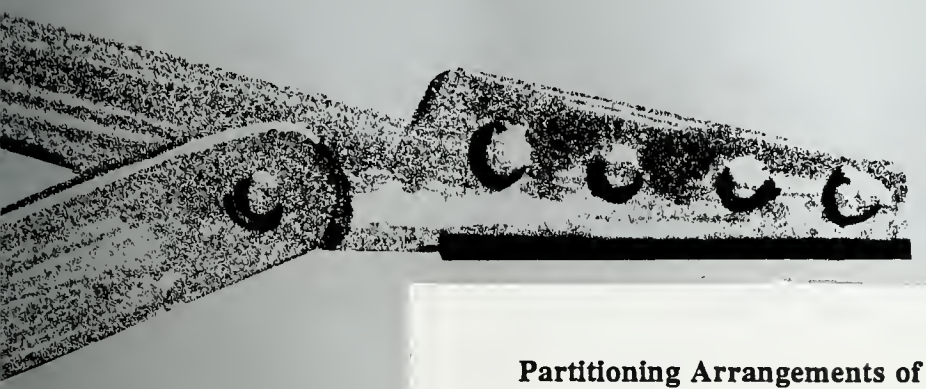
Robotics Research Technical Report



Partitioning Arrangements of Lines: I. An Efficient Deterministic Algorithm

by

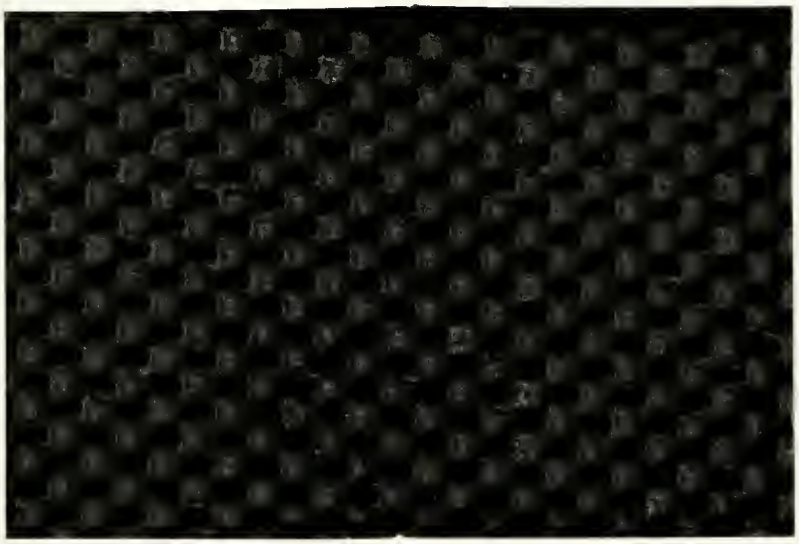
Pankaj K. Agarwal

Technical Report No. 446
Robotics Report No. 197
May, 1989

NYU COMPSCI TR-446
Agarwal, Pankaj K
Partitioning arrangements
of lines: I.
c.1

New York University
ınt Institute of Mathematical Sciences

Computer Science Division
251 Mercer Street New York, N.Y. 10012



Partitioning Arrangements of Lines: I. An Efficient Deterministic Algorithm

by

Pankaj K. Agarwal

Technical Report No. 446
Robotics Report No. 197
May, 1989

New York University
Dept. of Computer Science
Courant Institute of Mathematical Sciences
251 Mercer Street
New York, New York 10012

Work on this paper has been supported by Office of Naval Research Grant N00014-87-K-0129 National Science Foundation CER Grant DCR-83-20085, National Science Foundation Grant subcontract CMU-406349-55586, and by grants from the Digital Equipment Corporation and the IBM Corporation.

Partitioning Arrangements of Lines: I. An Efficient Deterministic Algorithm

Pankaj K. Agarwal[†]
Courant Institute of Mathematical Sciences
New York University
NY 10012

ABSTRACT

In this paper we consider the following problem: Given a set $\mathcal{L}$ of n lines in the plane, partition the plane into $O(r^2)$ triangles so that no triangle meets more than $O(\frac{n}{r})$ lines of $\mathcal{L}$. We present a deterministic algorithm for this problem with $O(nr \log n \log^{\omega} r)$ running time, where ω is a constant < 3.33.

[†]Work on this paper has been supported by Office of Naval Research Grant N00014-87-K-0129, by National Science Foundation Grant DCR-83-20085, and by grants from the Digital Equipment Corporation, and the IBM Corporation.

[†]A preliminary version of this paper appears in *Proceedings* 5th *Annual Symposium on Computational Geometry*, 1989, pp. 11-22.

1 Introduction

In the last few years several randomized divide and conquer algorithms for a variety of geometric problems have been developed ([AS], [Cla], [Clb], [Clc], [CS], [CTV], [EGS], [GOS] etc.), using ϵ-nets [HW] or the random sampling technique of [Clb] (see also [RSa], [RSb]). The ϵ-net theory shows that for a given set X of n objects and a set $\mathcal{R} \subseteq 2^X$ of *ranges* with finite *Vapnik-Chervonenkis dimension* (see [HW] for definition), there exists a subset $N \subset X$ of size r, for any $r > 0$, such that if $\tau \in \mathcal{R}$ and $\tau \cap N = \emptyset$, then $|\tau| = O(\frac{n}{r} \log r)$. Moreover, the theory asserts that a random sample N of size r will be an ϵ-net with high probability. A similar result has been independently obtained by Clarkson [Clb], [Clc]. It states roughly that if we draw a random sample N of size r from our set X and partition the underlying geometric space into cells with the property that (i) each cell can be defined in terms of only a "constant" number of elements of N, and (ii) each cell does not meet any element of N, then, with high probability, no cell meets more than $O(\frac{n}{r} \log r)$ elements of X. Random sampling techniques allow us to split the original problem into subproblems of small size, which can then be solved recursively. For example, if the objects are "lines in the plane" and an element of $\mathcal{R}$ is the "set of lines intersecting a given triangle", then these results imply that, for a given set $\mathcal{L}$ of n lines in the plane, there exists a subset $N \subset \mathcal{L}$ of size r such that any triangle which misses all lines in N intersects at most $O(\frac{n}{r} \log r)$ lines of $\mathcal{L}$. A problem involving the set $\mathcal{L}$ of lines can then be split into subproblems by computing the arrangement of the lines in N and by triangulating each face of the arrangement; each resulting triangle Δ induces a subproblem involving only $O(\frac{n}{r} \log r)$ lines of $\mathcal{L}$ intersecting Δ. As an example of a problem that benefits from this set-up, consider *Hopcroft's problem* (see [CSY]): Given m points and n lines in the plane, does any point lie on any line? Applying the above divide and conquer strategy to the given lines, and partitioning the points among the triangles, each subproblem involves only the points that fall within some triangle and the line cutting it. More details concerning this and other problems of this kind are given in the second part of this paper [Aga]. The same approach works for other geometric objects in the plane as well as in higher dimensions.

Unfortunately the random sampling technique or the ϵ-net theory only proves the existence of such a subset N, but does not show how to construct it deterministically (and efficiently). This has forced all algorithms based on these techniques to use randomization to obtain N (the only randomized step in most of these algorithms is that of choosing a good sample N). The randomization is justified because the above theories show that most subsets N of size r satisfy the desired properties, so a random choice of such a subset will succeed with high probability. Still, a major open problem in this area is to find an efficient deterministic algorithm to obtain a "good" sample. Recently Chazelle and Friedman [CF] gave a general framework which unifies the results of Haussler-Welzl [HW] and Clarkson [Clc], and yields a deterministic algorithm to construct a good sample. Although their algorithm runs in polynomial time, it is not very efficient (in particular its

time complexity is too high an overhead for most of the applications). This has motivated researchers to look for special cases, where faster deterministic constructions are possible. Woeginger [Wo] gave an $O(n)$ algorithm to construct an ϵ-net for a very special case, where the objects are points in the plane and the ranges are half-planes. A much more significant result in this direction has been obtained by Matoušek [Maa], who showed that, given a set of n lines in the plane, the plane can be partitioned into $O(r^2)$ triangles in time $O(nr^2 \log^2 r)$ so that no triangle intersects more than $O(\frac{n}{r})$ lines (however, these triangles in general are not formed by an arrangement of a sample subset of the lines). Note that an important property of the algorithm is that each triangle it produces is cut by only $O(\frac{n}{r})$ lines, rather than $O(\frac{n}{r} \log r)$, as promised by the probabilistic techniques. If r is constant, then this algorithm is optimal, and can be used to remove randomization from most of the "random-sampling" based algorithms involving lines or segments ([AS], [EGS], [GOS], [HW] etc.) However, it is not efficient if r is large, in particular if $r = n^\alpha$, for some $0 < \alpha < 1$. Let us be more specific concerning this efficiency issue. Even though Matoušek's algorithm is much faster than that of [CF], it is still inefficient when compared to the following "lower bound". Since the algorithm produces $O(r^2)$ triangles and each is met by $O(\frac{n}{r})$ lines, the total input size of all the corresponding subproblems (that is, the total number of line-triangle crossings) is $O(nr)$ (one can show that the bound is tight in the worst case). Thus it is natural to seek an improved algorithm whose complexity is close to $O(nr)$. As we will see later, this improvement does make a difference, because in many applications we do want to choose r to be n^α, for some $0 < \alpha < 1$.

In this paper we achieve such an improvement, and obtain an $O(nr \log n \log^\omega r)$ deterministic algorithm which, given a set $\mathcal{L}$ of n lines, partitions the plane into $O(r^2)$ triangles, none of which meets more than $O(\frac{n}{r})$ lines (here ω is some constant < 3.33). Then we apply this algorithm to remove randomization and to improve the time complexity of solutions to several problems. A common characteristic of the time complexity of previously known algorithms for these problems is that they are worse by a factor of $O(n^\delta)$, for any $\delta > 0$ (with a constant of proportionality depending on δ), than the worst case output size, or than some other natural measure of complexity that one can associate with the problem. For example, the running time of the best known (randomized) algorithm to compute incidences between m points and n lines is $O(m^{2/3-\delta} n^{2/3+2\delta} + (m+n)\log n)$, for any $\delta > 0$ [EGS], which is worse by a factor of $O((\frac{n^2}{m})^\delta)$ than the maximum possible number of such incidences. (A slightly faster but still randomized algorithm for this problem has been given in [EGH*] which runs in $O(m^{2/3} n^{2/3} \log^4 n + (m + n^{3/2}) \log^2 n)$ expected time. The difference between these two algorithms is that the first algorithm can be made deterministic without increasing its running time, using Matoušek's algorithm, while the second algorithm is not yet known to admit a similarly efficient determinization.) Factors like $O(n^\delta)$ appear in the bound of these algorithms due to the fact that they use small values of r (in most cases constant values). Although Matoušek's algorithm makes most of these algorithms deterministic, it also cannot choose a large value of r, because its running time is quadratic in terms of r, in which case the overhead of constructing the partition

dominates substantially the time complexity, resulting in an inefficient algorithm. On the other hand, by applying our algorithm, we can use a sufficiently large value of r, which allows us to remove the $O(n^\delta)$ factor from the bounds. Another disadvantage of using a small value of r is that the algorithm then becomes recursive and more complex. In contrast, by choosing an appropriate large value of r we partition the plane just once, because then the subproblems are sufficiently small, and can be solved directly by other means; this makes the algorithms much simpler.

In a companion paper [Aga] we describe several applications of our partitioning algorithm including the following ones:

(i) *Computing incidences between lines and points*: Given a set of n lines and a set of m points in the plane, compute how many lines pass through each point. (Alternatively, compute the lines passing through each point, or just determine whether any line passes through any point.) Edelsbrunner et al. [EGS] have given a randomized algorithm for this problem with expected running time $O(m^{2/3-\delta}n^{2/3+2\delta} + (m+n)\log n)$, for any $\delta > 0$. A slightly improved, but still randomized, algorithm has been given in [EGH*]. We present a deterministic algorithm with $O(m^{2/3}n^{2/3}\log^{2/3} n \log^{\omega/3} \frac{m}{\sqrt{n}} + (m+n)\log n)$ time complexity. Since the maximum number of incidences between m points and n lines is $\Theta(m^{2/3}n^{2/3} + m + n)$, our algorithm is close to optimal in the worst case.

(ii) *Computing many faces in an arrangement of lines*: Given a set of n lines and a set of m points in the plane, compute the faces in the arrangement of the lines containing the given points. Edelsbrunner et al. [EGS] have given a randomized algorithm for this problem with expected running time $O(m^{2/3-\delta}n^{2/3+2\delta} + n\log n \log m)$, for any $\delta > 0$. As in the case of the incidence problem, a slightly better randomized algorithm has been given in [EGH*]. We present a deterministic $O(m^{2/3}n^{2/3}\log^{5/3} n \log^{\omega/3} \frac{m}{\sqrt{n}} + n\log n)$ algorithm, again coming close to optimal in the worst case (see [CEGSW]).

(iii) *Computing many faces in an arrangement of segments*: This is the same problem as the previous one except that now we have a collection of segments instead of lines. The previous best solution is by Edelsbrunner et al. [EGS], which is randomized and has expected running time $O(m^{2/3-\delta}n^{2/3+2\delta} + n\alpha(n)\log^2 n \log m)$, for any $\delta > 0$, where $\alpha(n)$ is a functional inverse of Ackermann's function. We present a deterministic algorithm with improved time complexity $O(m^{2/3}n^{2/3}\log n \log^{\omega/3+1} \frac{n}{\sqrt{m}} + n\log^3 n)$.

(iv) *Counting segment intersections*: We give a deterministic $O(n^{4/3}\log^{(\omega+2)/3} n)$ algorithm to count the number of intersections in a given collection of n segments; this is an improvement over Guibas et al.'s algorithm [GOS], which counts the number of intersections in $O(n^{4/3+\delta})$ randomized expected time, for any $\delta > 0$.

(v) *Counting and reporting red-blue intersections*: Given a set Γ_r of n_r "red" segments

and another set Γ_b of n_b "blue" segments in the plane, count the number of intersections between Γ_r and Γ_b, or report all of them. (In this problem, we need to ignore the potentially large number of intersections within Γ_r or within Γ_b.) The previous best solution is by Agarwal and Sharir [AS], which reports all K red-blue intersections deterministically in $O((n_r\sqrt{n_b}+n_b\sqrt{n_r}+K)\log n)$ time, where $n = n_r + n_b$. We give a deterministic $O(n^{4/3}\log^{(\omega+2)/3} n)$ algorithm to count all red-blue intersections. It can report all K red-blue intersections in $O(n^{4/3}\log^{(\omega+2)/3} n + K)$ time.

(vi) *Implicit point location problem:* Given a collection of m points and a collection of (possibly intersecting) n triangles in the plane, find which points lie in the union of the triangles. This turns out to be a special case of a general problem of implicit point location of planar maps formed by overlapping figures. We present a deterministic algorithm with $O(m^{2/3}n^{2/3}\log^{2/3} n \log^{\omega/3} \frac{n}{\sqrt{m}} + (m+n)\log n)$ time complexity.

(vii) *Approximate half-plane range query*: Given a set S of n points in the plane and a parameter (not necessarily constant) $\epsilon > 0$, preprocess them so that, for any query line ℓ, we can approximately count the number of points lying above ℓ with an error of at most $\pm\epsilon n$. We give an algorithm that preprocesses S, in time $O(\frac{n}{\epsilon}\log n \log^{\omega}\frac{1}{\epsilon})$, into a data structure of size $O(\frac{n}{\epsilon})$, so that a query can be answered in $O(\log n)$ time.

(viii) *Constructing spanning trees with low stabbing number*: Given a set S of n points in the plane, we present an $O(n^{3/2}\log^{\omega+1} n)$ algorithm to construct a family of $k = O(\log n)$ spanning trees $\mathcal{T}_1, \ldots, \mathcal{T}_k$ of S with the property that, for any line ℓ there is tree $\mathcal{T}_i$, such that ℓ intersects at most $O(\sqrt{n})$ edges of $\mathcal{T}_i$. Moreover, with additional $O(n\log n)$ preprocessing and $O(n)$ space, the tree $\mathcal{T}_i$ corresponding to a query line ℓ can be determined in $O(\log n)$ time. The previous best known algorithm is by Matoušek [Mab], which runs in $O(n^{7/4}\log^2 n)$ time, and, moreover, produces a stabbing number $O(\sqrt{n}\log^2 n)$ instead of $O(\sqrt{n})$.

(ix) *Space query time tradeoff in simplex range searching*: Given a set S of n points in the plane and a parameter $n^{1+\epsilon_0} \le m \le n^{2-\epsilon_1}$, for some constants $\epsilon_0, \epsilon_1 > 0$, preprocess S so that for any query triangle, we can quickly compute the number of points contained in that triangle. We give an algorithm with $O(\frac{n}{\sqrt{m}}\log^{3/2} n)$ query time, using $O(m)$ space. The preprocessing time is bounded by $O(n\sqrt{m}\log^{\omega+1/2} n)$. Similar bounds have been obtained independently by Chazelle [Chc].

(x) *Overlapping planar maps:* Given two planar maps P, Q, and a bivariate function $F_P(x,y)$, $F_Q(x,y)$ associated with each of them, such that over each face of P the function F_P has some simple structure (e.g. it is constant, linear, or convex over each face), and similarly for Q, determine a point that minimizes $F_P(x,y) - F_Q(x,y)$. We show that if the maps satisfy certain conditions, then an optimal point can be computed in $O(n^{4/3}\log^{(\omega+2)/3} n)$ time, where n is the total complexity of the two maps. The details of this application can be found in [Agc].

Our algorithm for partitioning the plane works in two phases. The first phase partitions the plane into $O(r^2 \log^{\omega} r)$ triangles so that no triangle intersects more than $O(\frac{n}{r})$ lines. The second phase reduces the number of triangles to $O(r^2)$ till maintaining the property that no triangle meets more than $O(\frac{n}{r})$ lines, using a similar technique to that of Matoušek [Maa]. We believe that the second phase is redundant, i.e. that the first phase or some appropriate variant of it produces only $O(r^2)$ triangles, but we have not been able to prove it so far. Even if this is not the case, in most of the applications we can stop after the first phase and solve the subproblems directly within each of the resulting $O(r^2 \log^{\omega} r)$ triangles, without increasing the asymptotic complexity of the algorithm.

The paper is organized as follows. In Section 2 we describe the geometric concepts that we use in this paper. Section 3 gives an algorithm for a subproblem which is interesting in its own right, namely that of computing the κth leftmost intersection point induced by a set of n lines inside a given convex quadrilateral. Section 4 is the heart of our algorithm; it presents an algorithm to partition a given convex quadrilateral $\mathcal{Q}$ intersecting m lines and containing K of their intersection points into $O(\frac{m}{\zeta} + \frac{K}{\zeta^2})$ convex quadrilaterals (for an arbitrary parameter ζ) so that both the number of lines crossing any subquadrilateral $\mathcal{Q}'$ and the number of their intersections inside $\mathcal{Q}'$ are small (the precise conditions are given in Theorem 4.15). In Section 5 we describe the first phase of the partitioning algorithm, which basically consists of applying the algorithm of Section 4 recursively, and Section 6 describes the second phase of the algorithm. Section 7 shows how to modify the analysis to handle degenerate cases as well. In Section 8 we conclude with some final remarks and open problems.

2 Geometric preliminaries

This section defines the geometric concepts, and formalizes the notation that we will be using in this paper. Let $\mathcal{H} = \{\ell_1, \ell_2, \ldots, \ell_n\}$ denote a set of n lines in the plane. These lines induce a planar map called the *arrangement* $\mathcal{A}(\mathcal{H})$ of $\mathcal{H}$, whose *vertices* are the intersection points of lines in $\mathcal{H}$, *edges* are maximal connected portions of lines in $\mathcal{H}$ not containing a vertex, and *faces* are maximal connected portions of the plane not meeting any edge or vertex of $\mathcal{A}(\mathcal{H})$. See [Ed] for more details. For simplicity we assume that the lines in $\mathcal{H}$ are in general position, that is, no three lines are concurrent, and that no line in $\mathcal{H}$ is vertical. For any point $p \in \mathbb{R}^2$, we define its *level* to be the number of lines in $\mathcal{H}$ lying above it (not counting the lines passing through p). For any $0 \leq k < n$, the *k-level* of $\mathcal{A}(\mathcal{H})$ is the set of edges of $\mathcal{A}(\mathcal{H})$ whose level is k (see figure 1). A k-level of $\mathcal{A}(\mathcal{H})$ is a monotone polygonal chain with two unbounded rays. More details on k-levels can be found in [Ed], [EW], [Wea].

Lemma 2.1 *Let p_1 and p_2 be two points in the plane whose levels are k_1 and k_2 respectively. Then the line segment $\overline{p_1 p_2}$ intersects at least $|k_1 - k_2|$ lines of $\mathcal{H}$.*

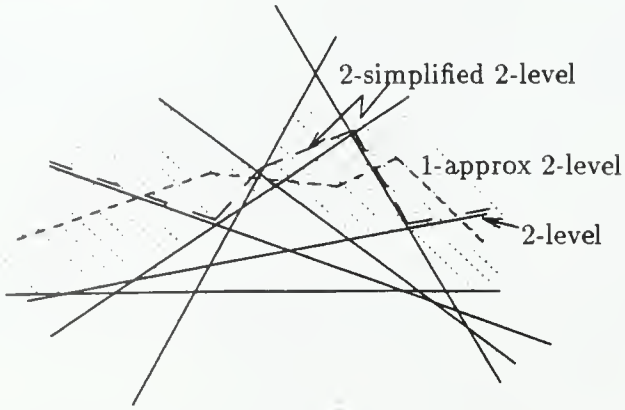


Figure 1: 2-level, 1-approximate 2-level, 2-simplified 2-level

Proof: Obvious from the definition of k-levels.

□

Lemma 2.2 ([Maa]) *Let $\mathcal{U}_1, \mathcal{U}_2, \ldots, \mathcal{U}_s$ denote pairwise disjoint sets of levels of $\mathcal{A}(\mathcal{L})$ with $|\mathcal{U}_i| \geq \omega$ for all $i \leq s$, then there exists a level $U_i \in \mathcal{U}_i$ such that $\sum_{i=1}^{s} |U_i| \leq \frac{n^2}{2\omega}$.*

□

We call an x-monotone polygonal path Π (not necessarily formed by the edges of $\mathcal{A}(\mathcal{H})$) an *ϵ-approximate k-level*, for $\epsilon \leq k$, if it lies in the strip bounded between the $k - \epsilon$ and $k + \epsilon$ levels of $\mathcal{A}(\mathcal{H})$ (see figure 1). A set of ϵ-approximate $2\epsilon i$-levels, for $i \leq \lfloor \frac{n}{2\epsilon} \rfloor$, is called an *$\epsilon$-approximate leveling* of $\mathcal{A}(\mathcal{H})$. Let $p_0, p_1, \ldots, p_m$ be the vertices of a k-level of $\mathcal{A}(\mathcal{H})$. For any $\delta < m$, we define the *δ-simplified k-level* to be the polygonal path formed by connecting p_0 to p_δ, p_δ to $p_{2\delta}$, $\ldots$, $p_{\lfloor \frac{m}{\delta} \rfloor \delta}$ to p_m, and concatenated with the left and right rays of the k-level incident to p_1 and p_m respectively. Edelsbrunner and Welzl [EW] proved that

Lemma 2.3 ([EW]) *For a given set of n lines $\mathcal{H}$, and $0 \leq k < n$, a δ-simplified k-level is a $\lceil \frac{\delta}{2} \rceil$-approximate k-level of $\mathcal{A}(\mathcal{H})$.*

□

Let $\mathcal{Q}$ be a convex quadrilateral with vertices q_1, q_2, q_3, q_4 ordered in counter-clockwise direction, and let $\partial\mathcal{Q}$ denote the boundary of $\mathcal{Q}$. Let $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_m\}$ denote a set of lines passing through $\mathcal{Q}$, and let K be the number of intersection points of $\mathcal{L}$ contained inside $\mathcal{Q}$. The intersection points of $\partial\mathcal{Q}$ and $\mathcal{L}$ are called the *endpoints* of $\mathcal{L}$. For simplicity let us assume that every line of $\mathcal{L}$ intersects $\partial\mathcal{Q}$ at two points, i.e. does not touch $\mathcal{Q}$ at one of its vertices (or at an edge).

The set $\mathcal{L}$ of lines can be partitioned into three different classes according to the location of their endpoints (see figure 2).

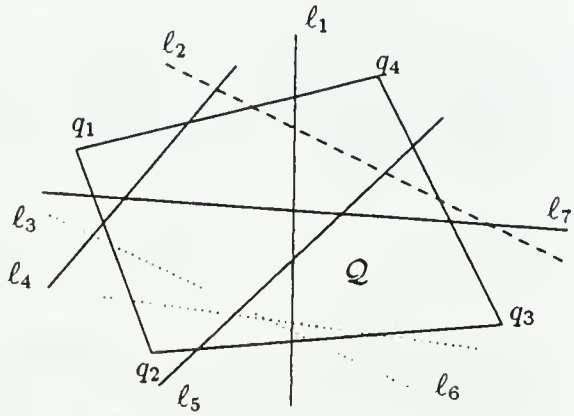


Figure 2: $\mathcal{Q}$ and $\mathcal{L}$: solid lines denote red lines, dashed lines denote green lines and dotted lines denote blue lines.

(i) "Red" lines $\mathcal{L}_r$: a line ℓ belongs to $\mathcal{L}_r$ if one of its endpoints lies on $\overline{q_1q_2}$ or $\overline{q_2q_3}$ (referred to as the left endpoint) and the other one lies on $\overline{q_1q_4}$ or $\overline{q_3q_4}$ (referred to as the right endpoint), i.e. ℓ crosses the diagonal $\overline{q_1q_3}$; let $|\mathcal{L}_r| = m_r$.

(ii) "Blue" lines $\mathcal{L}_b$: a line ℓ belongs to $\mathcal{L}_b$ if one of its endpoints lies on $\overline{q_1q_2}$ (referred to as the left endpoint) and the other one lies on $\overline{q_2q_3}$ (referred to as the right endpoint); let $|\mathcal{L}_b| = m_b$.

(iii) "Green" lines $\mathcal{L}_g$: a line ℓ belongs to $\mathcal{L}_g$ if one of its endpoints lies on $\overline{q_1q_4}$ (referred to as the left endpoint) and the other one lies on $\overline{q_4q_3}$ (referred to as the right endpoint); let $|\mathcal{L}_g| = m_g$.

These three sets are pairwise disjoint and $\mathcal{L}_r \cup \mathcal{L}_g \cup \mathcal{L}_b = \mathcal{L}$. Let K_{xy} denote the number of intersection points between $\mathcal{L}_x$ and $\mathcal{L}_y$ lying inside $\mathcal{Q}$, where $x, y \in \{r, b, g\}$. It is easily seen that $\mathcal{L}_b$ and $\mathcal{L}_g$ do not intersect inside $\mathcal{Q}$, that is $K_{bg} = 0$. Without loss of generality we can assume that $m_r \geq \frac{m}{2}$ because otherwise we can cyclically renumber the vertices of $\mathcal{Q}$ so that the above inequality is satisfied. For the sake of exposition we assume that q_1 is the upper left corner of $\mathcal{Q}$.

Let $A = \{a_1, \ldots, a_{m_r}\}$ (resp. $B = \{b_1, \ldots, b_{m_r}\}$) be the left (resp. right) endpoints of $\mathcal{L}_r$ appearing in counter-clockwise (resp. clockwise) direction around $\partial\mathcal{Q}$. For each line $\ell_i \in \mathcal{L}_r$ let $a_{\pi(i)}$ (resp. $b_{\sigma(i)}$) denote its left (resp. right) endpoint.

3 Selecting the κth leftmost intersection point

Let $p_1, \ldots, p_K$ denote the intersection points of the lines in $\mathcal{L}$ lying inside $\mathcal{Q}$ and ordered in increasing x-coordinate. In this section we consider the problem of calculating the κth leftmost point p_κ of these intersections, for any given $\kappa \leq K$. The algorithm extends the

recent algorithm of Cole et al. [CSSS] for calculating the kth leftmost intersection point of n lines in the entire plane. One of the key steps of the algorithm is to count the number of intersection points of $\mathcal{L}$ lying inside a given convex quadrilateral. We first give an algorithm that counts the number of intersections inside a convex quadrilateral in time $O(m \log m)$, where m is the number of lines passing through $\mathcal{Q}$.

3.1 Counting intersection points inside a convex quadrilateral

We partition the set of lines $\mathcal{L}$ into $\mathcal{L}_r$, $\mathcal{L}_b$ and $\mathcal{L}_g$ as defined earlier, and count each of K_{xy}, for $x, y \in \{r, b, g\}$, separately. First, we show how to count the number of red-red intersection points lying inside $\mathcal{Q}$, i.e. K_{rr}. It is obvious that two red lines ℓ_i and ℓ_j in $\mathcal{L}$ intersect if and only if $\pi(i) < \pi(j)$ and $\sigma(i) > \sigma(j)$, or vice versa. We thus obtain K_{rr} by counting the number of inversions in σ with respect to π, which is easily done in time $O(m_r \log m_r)$ by sorting the endpoints of $\mathcal{L}_r$ along the boundary of $\mathcal{Q}$. Similarly, we can compute K_{bb} (resp. K_{gg}) in time $O(m_b \log m_b)$ (resp. $O(m_g \log m_g)$).

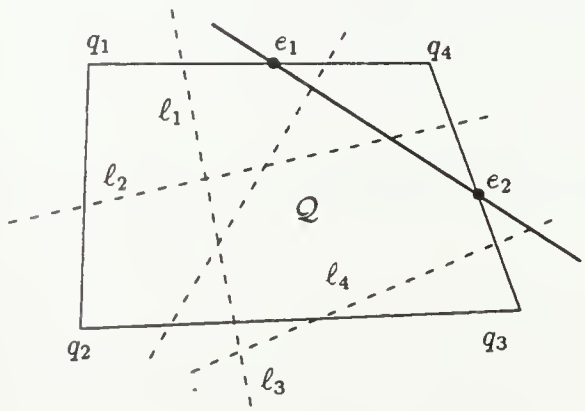


Figure 3: Green line ℓ and its endpoints e_1, e_2; ℓ_1, ℓ_4 do not intersect ℓ

As for determining K_{rg}, let e_1 (resp. e_2) be the left (resp. right) endpoint of a green line ℓ. The line ℓ intersects a red line ℓ_j if the right endpoint of ℓ_j lies on either $\overline{e_1 q_4}$ or $\overline{q_4 e_2}$ (see figure 3). Therefore using the same sorted list B, we can compute the number of red lines that ℓ intersects by locating e_1 and e_2 in B, which implies that K_{rg} can be computed in time $O(m_g \log m_r)$. Similarly, K_{rb} can be computed in time $O(m_b \log m_r)$. Thus, we have shown

Lemma 3.1 *For a given convex quadrilateral $\mathcal{Q}$ and a set $\mathcal{L}$ of m lines passing through it, we can count the number of intersections of $\mathcal{L}$ lying inside $\mathcal{Q}$ in time $O(m \log m)$.*

□

Remark 3.2: The above algorithm can be extended to count the number of intersections lying inside any k-gon $\mathcal{B}$ in $O(km \log m)$ time, by triangulating $\mathcal{B}$, and by counting the number of intersections in each triangle separately.

Next we give an algorithm for finding the κth leftmost intersection point inside $\mathcal{Q}$ using the counting procedure just described.

3.2 Computing the κth leftmost intersection

Cole et al. [CSSS] have given an $O(m \log m)$ algorithm that returns the κth leftmost intersection point in a set $\mathcal{L}$ of m lines in the plane. Their algorithm cannot be applied directly to our case because it counts all intersection points of $\mathcal{L}$ lying to the left of the κth one, while we count only those intersection points that lie inside $\mathcal{Q}$. However, we will show that the algorithm of [CSSS] can be easily adapted to our case. We assume that the reader is familiar with this algorithm, so we describe it here only briefly, and show how to modify it for our case; for more details, see [CSSS].

Let $\pi(a)$ denote the permutation of lines in $\mathcal{L}$ sorted in decreasing order of their intercepts with the line $x = a$, assuming that no two lines in $\mathcal{L}$ intersect at $x = a$. The *rank* of a, denoted by $\psi(a)$, is defined to be the number of intersection points of $\mathcal{L}$ lying inside $\mathcal{Q}$ and to the left of $x = a$. The κth leftmost intersection point p_κ is such that $\psi(a^*) = \kappa$, where $a^* = x(p_\kappa) + \epsilon$, for some sufficiently small $\epsilon > 0$ (in this description we assume that no two points p_i, p_j have the same x-coordinate).

Cole et al.'s algorithm [CSSS] is based on Megiddo's technique [Me], further improved by Cole [Co], for implicit parametric searching. Using a sorting network, e.g. that of Ajtai et al. [AKS], $\pi(a^*)$ can be obtained by performing $O(m \log m)$ comparisons, performed in $O(\log m)$ batches of $O(m)$ independent comparisons each, where each comparison answers a question of the form: "Does ℓ_i lie above ℓ_j at $x = a^*$?". Although we do not know a^*, we can answer a question of this form by computing the rank of the abscissa $x(q_{ij})$, where q_{ij} is the intersection point of ℓ_i and ℓ_j. If $\psi(x(q_{ij})) < \kappa$, then q_{ij} lies to the left of $x = a^*$, so the order of ℓ_i and ℓ_j there coincides with their order at $x = -\infty$; otherwise the order is reversed. In such a sorting network there are $O(\log m)$ levels and at each level $s = O(m)$ questions are asked; let $z_1, z_2, \ldots, z_s$ denote the x-coordinates of the intersection points corresponding to the questions to be answered at a level j. If we resolve the question for the median value z_{med} of $z_1, \ldots, z_s$, then we can resolve the question for at least half of the above points, e.g. if $z_{\text{med}} < a^*$, then $z_i < a^*$ for all $z_i < z_{\text{med}}$. For the remaining half of the points, the question can be resolved by applying the same method recursively, thus allowing us to resolve all m questions by actually computing the rank of only $O(\log m)$ points. Once all comparisons are resolved at level j, we can continue to level $j + 1$ of the network. At the end, all comparisons have been resolved and we have obtained the sorted

order of the lines in $\mathcal{L}$ along $x = a^*$, still without knowing the exact value of a^*. However, this order is easily seen to determine uniquely how many intersection points of $\mathcal{L}$ lie to the left of $x = a^*$, thus the set of these intersections, as well as the subset of these points lying inside $\mathcal{Q}$, are fully determined at the end of the algorithm. By keeping track of all inequalities of the form $z_{\text{med}} < a^*$ or $z_{\text{med}} > a^*$ we finally obtain an interval $\alpha < a^* < \beta$, and the logic of the procedure is easily seen to imply that α is the x-coordinate of p_κ. The point p_κ itself is also easy to obtain by recording, for each z_{med}, the two lines that induce it.

To resolve a comparison at $x = z_{\text{med}}$, we proceed as follows. Let $\mathcal{Q}^-$ denote the portion of $\mathcal{Q}$ lying to the left of the vertical line $x = z_{\text{med}}$. by Lemma 3.1 and the subsequent Remark, we can count the number of intersection points of $\mathcal{L}$ lying inside $\mathcal{Q}^-$ (which is at most a pentagon) in $O(m \log m)$ time. Thus the rank of z_{med} can be computed in $O(m \log m)$ time, so the overall running time of the algorithm is $O(m \log^3 m)$. By using Cole's improvement [Co], the time can be reduced to $O(m \log^2 m)$. Cole et al. [CSSS] observed that it is not necessary to compute the exact rank of a point because we are only interested in knowing whether z_{med} lies to the left or to the right of p_κ. They proved that it suffices to compute an approximate rank of a point, which can be done in $O(m)$ amortized time, yielding an $O(m \log m)$ algorithm. Using the same idea we can also compute our version of approximate rank in amortized time $O(m)$, and therefore we obtain

Theorem 3.3 *Given a convex quadrilateral $\mathcal{Q}$, a set of m lines $\mathcal{L}$ and an integer κ, we can determine, in time $O(m \log m)$, a vertical line, so that $\mathcal{Q}$ contains exactly κ intersection points to the left of that line.*

□

Remark 3.4:

(i) The remark following Lemma 3.1 implies that we can extend the above algorithm to any k-gon with $O(km \log m)$ running time.

(ii) The above algorithm uses the sorting network of Ajtai et al. [AKS], which involves a very large constant, therefore making our algorithm impractical. A possible solution to circumvent this problem is to use the much simpler Batcher's sorting network [Ba]; this network has depth $O(\log^2 m)$ and therefore the overall running time of the resulting algorithm becomes $O(m \log^2 m)$. However, to obtain a better asymptotic complexity, we will be using the former algorithm.

4 Partitioning a convex quadrilateral

This section describes an algorithm to partition the convex quadrilateral $\mathcal{Q}$ into convex quadrilateral subcells so that on the average only few lines pass through each cell, and the

maximum number of lines passing through each cell is also low. To be more precise, for any given positive integer $\zeta < m$, we want to partition $\mathcal{Q}$ into $M = O(\frac{m}{\zeta} + \frac{K}{\zeta^2})$ cells so that the following conditions are satisfied (where m_i is the number of lines passing through the ith cell, and K_i is the number of intersections within the ith cell):

$$\text{(i)} \quad \sum_{i=1}^{M} m_i = O\left(m + \frac{K}{\zeta}\right), \qquad \max_i m_i = O(\sqrt{K})$$

$$\text{(ii)} \quad \sum_{i=1}^{M} K_i = K, \qquad \max_i K_i \leq \zeta\sqrt{K}.$$

4.1 A special case

Before solving this general problem we consider a special case in which $\mathcal{Q}$ contains only "red" lines and we do not care how many intersection points lie in a cell. Moreover, although most cells in the construction to follow will be quadrilaterals, some could have up to six edges.

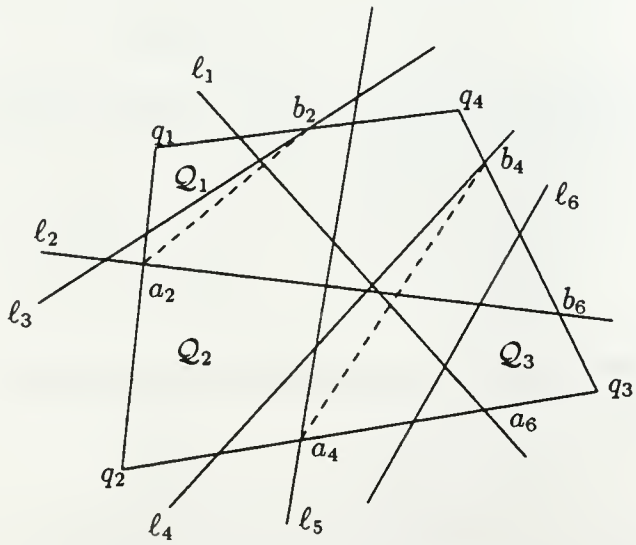


Figure 4: Partitioning $\mathcal{Q}$ into $\mathcal{Q}_1, \ldots, \mathcal{Q}_t$, with $m = 6$ and $\zeta = 2$. The dashed edges are pseudo edges.

First, compute the left and right endpoints of all lines in $\mathcal{L}$ and sort them to obtain the lists A and B. Next, for all $1 \leq i < t = \lceil \frac{m}{\zeta} \rceil$ connect $a_{i\zeta}$ to $b_{i\zeta}$. The segments $\overline{a_{i\zeta} b_{i\zeta}}$ are called *pseudo edges* (see figure 4). The pseudo edges partition $\mathcal{Q}$ into cells $\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_t$. The following sequence of lemmas show that this partitioning has the desired properties.

Lemma 4.1 *A line $\ell_i \in \mathcal{L}$ intersects at least $|\pi(i) - \sigma(i)|$ lines of $\mathcal{L}$ inside $\mathcal{Q}$.*

Proof: Without loss of generality assume that $\pi(i) \geq \sigma(i)$. Since only $\sigma(i) - 1$ lines have their right endpoints preceding the right endpoint of ℓ_i, at least $\pi(i) - \sigma(i)$ of the lines, whose left endpoints precede that of ℓ_i, have their right endpoints after $b_{\sigma(i)}$ in B. But every such line crosses the line ℓ_i inside $\mathcal{Q}$, showing that at least $|\pi(i) - \sigma(i)|$ lines of $\mathcal{L}$ intersect ℓ_i inside $\mathcal{Q}$.

□

Let δ_i denote the number of pseudo edges intersected by the line ℓ_i.

Lemma 4.2 $$\sum_{i=1}^{m} \delta_i \leq \frac{2K}{\zeta} + m.$$

Proof: If a line ℓ_i intersects δ_i pseudo edges, then it is easily seen that

$$|\pi(i) - \sigma(i)| \geq (\delta_i - 1)\zeta.$$

Summing over all lines, we obtain

$$\sum_{i=1}^{m} |\pi(i) - \sigma(i)| \geq \sum_{i=1}^{m} (\delta_i - 1)\zeta$$

Let ν_i denote the number of lines of $\mathcal{L}$ that ℓ_i intersects inside $\mathcal{Q}$. We have $\sum_{i=1}^{m} \nu_i = 2K$, because there are only K intersection points inside $\mathcal{Q}$ and each intersection is counted twice. By Lemma 4.1, $|\pi(i) - \sigma(i)| \leq \nu_i$, therefore

$$\sum_{i=1}^{m} (\delta_i - 1)\zeta \leq 2K \quad \text{or} \quad \sum_{i=1}^{m} \delta_i \leq \frac{2K}{\zeta} + m.$$

□

Let m_i denote the number of lines passing through the cell $\mathcal{Q}_i$. Since the line ℓ_j lies in $\delta_j + 1$ cells, we have

Corollary 4.3 $$\sum_{i=1}^{t} m_i = \sum_{j=1}^{m} (\delta_j + 1) \leq 2\left(\frac{K}{\zeta} + m\right)$$ □

Lemma 4.4 *Let μ_i denote the number of lines passing through the cell $\mathcal{Q}_i$ but not having any of their endpoints on $\partial\mathcal{Q}_i$. Then $\mu_i \leq 2\sqrt{K}$, for all $i \leq t$.*

Proof: Let $\mathcal{L}_i$ denote the set of lines passing through the cell $\mathcal{Q}_i$ but not having any endpoint in $\mathcal{Q}_i$. Partition the set $\mathcal{L}_i$ into disjoint subsets $\mathcal{L}_{xy}$, for $x = 1, \ldots, t - i$, $y = 1, \ldots, i - 1$, such that a line ℓ_j belongs to $\mathcal{L}_{xy}$ if one of its endpoint lies in $\mathcal{Q}_{i+x}$ and the other in $\mathcal{Q}_{i-y}$. Let $\mu_{xy} = |\mathcal{L}_{xy}|$. Note that

$$\forall \ell_j \in \mathcal{L}_{xy},\ |\pi(j) - \sigma(j)| \geq \zeta(x + y - 1).$$

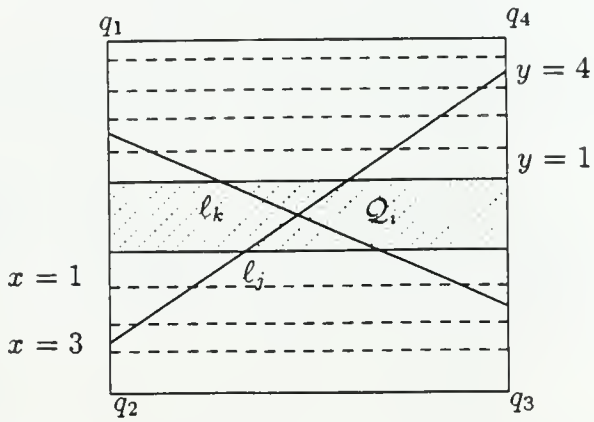


Figure 5: ℓ_j intersects at least 6ζ lines of $\mathcal{L}$.

By Lemma 4.1 a line $\ell_j \in \mathcal{L}_{xy}$ intersects at least $(x + y - 1)\zeta$ lines (see figure 5). Summing over all x and y we get

$$\begin{aligned} 2K &\geq \sum_x \sum_y \mu_{xy}(x + y - 1) \cdot \zeta \\ \text{or} \quad \frac{2K}{\zeta} &\geq \sum_x \sum_y x \cdot \mu_{xy} + \sum_x \sum_y y \cdot \mu_{xy} - \sum_x \sum_y \mu_{xy}. \end{aligned} \tag{4.1}$$

Let $\mathcal{L}'_x \subseteq \mathcal{L}_i$ denote the set of all lines having one endpoint in $\mathcal{Q}_{i+x}$. Then

$$\sum_x \sum_y x \cdot \mu_{xy} = \sum_x x\Big(\sum_y \mu_{xy}\Big) = \sum_x x|\mathcal{L}'_x|. \tag{4.2}$$

Similarly, if $\mathcal{L}''_y \subseteq \mathcal{L}_i$ denotes the set of lines having one endpoint in $\mathcal{Q}_{i-y}$, then

$$\sum_x \sum_y y \cdot \mu_{xy} = \sum_y y|\mathcal{L}''_y|.$$

Moreover $\sum_x \sum_y \mu_{xy} = \mu_i$. Therefore (4.1) can be written as

$$\sum_x x \cdot |\mathcal{L}'_x| + \sum_y y \cdot |\mathcal{L}''_y| - \mu_i \leq \frac{2K}{\zeta} \tag{4.3}$$

Since $\sum_x |\mathcal{L}'_x| = \mu_i$ and for all x, $|\mathcal{L}'_x| \leq 2\zeta$, the term $\sum_x x|\mathcal{L}'_x|$ minimizes when

$$|\mathcal{L}'_x| = \begin{cases} 2\zeta & \text{for } 1 \leq x \leq \left\lfloor \frac{\mu_i}{2\zeta} \right\rfloor \\ \mu_i - 2\zeta \left\lfloor \frac{\mu_i}{2\zeta} \right\rfloor & \text{for } x = \left\lfloor \frac{\mu_i}{2\zeta} \right\rfloor + 1 \\ 0 & \text{otherwise} \end{cases}$$

Hence,

$$\begin{aligned}\sum_x x \cdot |\mathcal{L}'_x| &\geq \sum_{x=1}^{\lfloor \frac{\mu_i}{2\zeta} \rfloor} x \cdot 2\zeta + \left(\mu_i - 2\zeta\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor\right)\left(\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor + 1\right) \\ &\geq \frac{2\zeta}{2}\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor\left(\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor + 1\right) + \left(\mu_i - 2\zeta\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor\right)\left(\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor + 1\right) \\ &= \left(\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor + 1\right)\left(\mu_i - \zeta\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor\right) \qquad (4.4)\end{aligned}$$

Similarly,

$$\sum_y y \cdot |\mathcal{L}''_y| \geq \left(\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor + 1\right)\left(\mu_i - \zeta\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor\right) \qquad (4.5)$$

By substituting (4.4) and (4.5) in (4.3), we obtain

$$\frac{2K}{\zeta} \geq 2\left(\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor + 1\right)\left(\mu_i - \zeta\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor\right) - \mu_i \qquad (4.6)$$

Let $\left\lfloor\frac{\mu_i}{2\zeta}\right\rfloor = a$ and $\frac{\mu_i}{2\zeta} = a + \nu$, for some $0 \leq \nu < 1$, then (4.6) becomes

$$\begin{aligned}\frac{2K}{\zeta} &\geq 2(a+1)(2\zeta(a+\nu) - \zeta a) - 2\zeta(a+\nu) \\ \text{or} \quad \frac{K}{\zeta^2} &\geq (a+1)(a+2\nu) - (a+\nu) \\ &= a^2 + 2a\nu + \nu \\ &= (a+\nu)^2 + \nu(1-\nu) \\ &\geq (a+\nu)^2 \qquad \text{(because } 0 \leq \nu < 1\text{)} \\ &= \frac{\mu_i^2}{4\zeta^2}\end{aligned}$$

Therefore, $\mu_i \leq 2\sqrt{K}$.

□

Concerning running time, notice that, the sets A and B, and therefore $\mathcal{Q}_1, \ldots, \mathcal{Q}_t$ can be computed in time $O(m \log m)$. Moreover, for each line ℓ_j we can easily compute the cells it crosses— they are simply all the cells lying between (and including) the cell containing $a_{\pi(j)}$ and the cell containing $b_{\sigma(j)}$. Thus the total time spent in computing the lines passing through each cell is bounded by $O(m \log m + K/\zeta)$. Now it follows from Theorem 3.3 that number K_i of intersections within $\mathcal{Q}_i$ can be computed in time $O(m_i \log m_i)$. Therefore the total time spent in computing intersections is at most

$$O\Big(\sum_i m_i \log m_i\Big) = O\Big((\sum_i m_i)\log m\Big) = O\Big((m + \frac{K}{\zeta})\log m\Big).$$

Hence, we can conclude that

Lemma 4.5 *The above algorithm partitions $\mathcal{Q}$ into $t = \lceil \frac{m}{\zeta} \rceil$ subcells $\mathcal{Q}_1, \ldots, \mathcal{Q}_t$ in time $O(m \log m)$ such that (i) $\sum_{i=1}^{t} m_i \leq 2\left(m + \frac{K}{\zeta}\right)$, and (ii) $\max_i m_i \leq 2\sqrt{K} + 2\zeta$. Moreover, we can compute $\mathcal{L}_i$ and K_i (as defined above) for all $\mathcal{Q}_i$, in time $O((m + \frac{K}{\zeta}) \log m)$.*

□

Remark 4.6: Note that in the special case at hand we have ignored the issue of making the numbers K_i small. Also, the resulting partitioning can have one cell with 6 edges or two cells with 5 edges each. These issues will be addressed in the following general algorithm.

4.2 General Algorithm

Next we give an algorithm for the general case using the above procedure as a subroutine. Our algorithm consists of five steps.

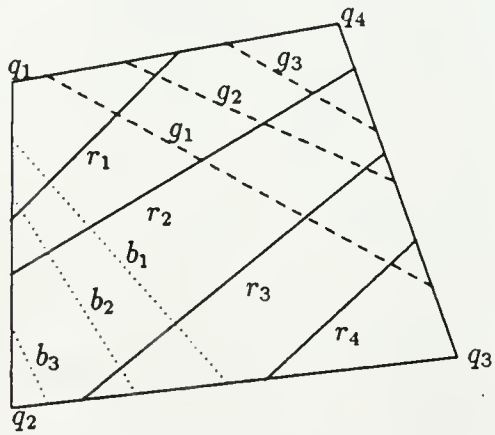


Figure 6: Red, green and blue pseudo edges

The first step applies the preceding algorithm to the collection of red lines, to partition the quadrilateral $\mathcal{Q}$ into a collection of cells $\mathbf{Q}^r = \{\mathcal{Q}_1^r, \ldots, \mathcal{Q}_t^r\}$, for $t = \lceil \frac{m_r}{\zeta} \rceil$, so that every cell contains the left (resp. right) endpoints of at most ζ red lines. Let $R = \{r_1, \ldots, r_{t-1}\}$ denote the set of pseudo edges that bound the quadrilaterals of $\mathbf{Q}^r$; they will be referred to as "red" pseudo edges.

The second step applies the preceding algorithm to the collection of green lines (where this time $\partial\mathcal{Q}$ is divided into a left and a right portions at q_4 and q_2), to partition $\mathcal{Q}$ into a collection of cells $\mathbf{Q}^g = \{\mathcal{Q}_1^g, \ldots, \mathcal{Q}_u^g\}$, for $u = \lceil \frac{m_g}{\zeta} \rceil$, so that every cell contains the left (resp. right) endpoints of at most ζ green lines. Let $G = \{g_1, \ldots, g_{u-1}\}$ denote the set of pseudo edges bounding the quadrilaterals of $\mathbf{Q}^g$; these pseudo edges will be referred to as "green" pseudo edges. Since the endpoints of all green lines lie on $\overline{q_1 q_4}$ and $\overline{q_3 q_4}$, all green pseudo edges extend from $\overline{q_1 q_4}$ to $\overline{q_3 q_4}$. Assume that the quadrilaterals $\mathcal{Q}_i^g$ are sorted

in their order along $\overrightarrow{q_1q_4}$, and that $\mathcal{Q}_i^g$ is bounded by the two pseudo edges g_{i-1}, g_i, for $1 < i < u$.

In the third step we apply the same algorithm to the collection of blue lines (again, partitioning $\partial\mathcal{Q}$ at q_2 and q_4), to partition $\mathcal{Q}$ into a collection of cells $\mathbf{Q}^b = \{\mathcal{Q}_1^b, \ldots, \mathcal{Q}_v^b\}$, for $v = \lceil \frac{m_b}{\zeta} \rceil$, so that every cell contains the left (resp. right) endpoints of at most ζ blue lines. Let $B = \{b_1, \ldots, b_{v-1}\}$ denote the set of pseudo edges bounding the quadrilaterals of $\mathbf{Q}^b$; they are referred to as "blue" pseudo edges. This time all pseudo edges extend from $\overline{q_1q_2}$ to $\overline{q_2q_3}$. Assume that the quadrilaterals $\mathcal{Q}_i^b$ are sorted in their order along $\overrightarrow{q_1q_2}$, and that $\mathcal{Q}_i^b$ is bounded by the two pseudo edges b_{i-1}, b_i, for $1 < i < v$. See figure 6 for an illustration of the resulting pseudo edges.

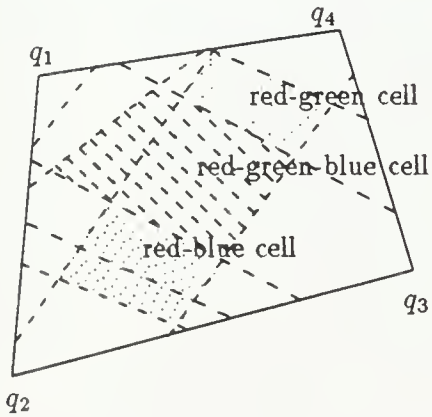


Figure 7: Internal and boundary cells: shaded regions denote internal cells

Let $\mathcal{S} = R \cup B \cup G$, and $\mathcal{E}$ the set of endpoints of segments in $\mathcal{S}$. Let $\mathbf{F}$ denote the set of bounded faces in the arrangement of $\partial\mathcal{Q} \cup \mathcal{S}$. If a face $\mathcal{F}_\alpha \in \mathbf{F}$ touches the boundary of $\mathcal{Q}$, then it is called a *boundary cell*, otherwise it is called an *internal cell* (see figure 7). The boundary of an internal cell is formed by four pseudo edges, two of which are always red. On the basis of the color of their edges we partition the cells into three categories:

(i) red-green cells: cells formed by the intersection of $\mathcal{Q}_i^g$ for $1 < i \leq u$ and $\mathcal{Q}_j^r$ for $1 \leq j \leq t$. A red-green cell formed by $\mathcal{Q}_i^g \cap \mathcal{Q}_j^r$ is denoted by $\mathcal{F}_{ij}$. If $\mathcal{F}_{ij}$ is an internal cell, then its edges are portions of $g_{i-1}, g_i \in G$ and $r_{j-1}, r_j \in R$ (and $i < u$, $1 < j < t$).

(ii) red-blue cells: cells formed by the intersection of $\mathcal{Q}_i^b$ for $1 < i \leq v$ and $\mathcal{Q}_j^r$ for $1 \leq j \leq t$. A red-blue cell formed by $\mathcal{Q}_i^b \cap \mathcal{Q}_j^r$ is denoted by $\mathcal{F}_{ij}$. If $\mathcal{F}_{ij}$ is an internal cell, then its edges are portions of $b_{i-1}, b_i \in B$ and $r_{j-1}, r_j \in R$ (and $i < v$, $1 < j < t$).

(iii) red-blue-green cells: cells formed by the intersection of $\mathcal{Q}_1^g$, $\mathcal{Q}_1^b$ and $\mathcal{Q}_i^r$ for $1 \le i \le t$. We denote such a cell by $\mathcal{F}_{1i}$. If $\mathcal{F}_{1i}$ is an internal cell, then its edges are portions of b_1, g_1, r_{i-1} and r_i (and $1 < i < t$).

The problem with the boundary cells produced by the arrangement of $\partial\mathcal{Q} \cup \mathcal{S}$ is that some of them may have more than four edges. However, since every cell has at most four pseudo edges, it cannot have more than eight edges in total. At the fourth step of our algorithm, we partition all boundary cells with more than four edges into two or three convex quadrilaterals and triangles by appropriate diagonals, which we also call pseudo edges (see figure 8).

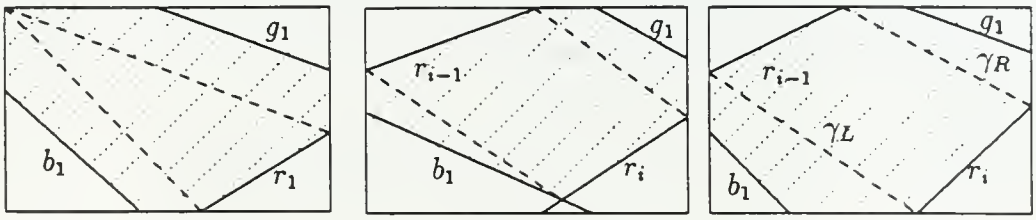


Figure 8: Cells with seven or eight edges: dashed lines are the pseudo edges added to partition the cell into convex quadrilaterals and triangles

A cell with five or six edges is partitioned arbitrarily into two convex quadrilaterals, but a cell with seven or eight edges is partitioned in such a way so that one of the new pseudo edges intersects only red and green lines, and the other intersects only red and blue lines. It is easily seen that such a partition is always possible. For example, an 8-cell lying between the red pseudo edges r_{i-1}, r_i is partitioned by connecting the left endpoint of r_{i-1} to the left endpoint of r_i, and the right endpoint of r_{i-1} to the right endpoint of r_i (see figure 8). 7-cells are partitioned in a similar manner.

Note that there can be at most one cell having 7 or 8 edges. The existence of an 8-cell implies that no pair of pseudo edges intersect, and the existence of a 7-cell allows only very limited pattern of pseudo-edge intersections. Moreover the existence of a 7 or 8-cell implies that the first step creates at least one red pseudo edge.

Next, for each of the cells produced so far, we compute the lines passing through it, and count the number of intersection points lying in it. If $K \ge 6\zeta^2$ and a cell $\mathcal{F}_\alpha$ has $K_\alpha > \zeta\sqrt{K}$ intersection points, then we partition it into $\lceil \frac{K_\alpha}{\zeta\sqrt{K}} \rceil$ cells, each containing at most $\zeta\sqrt{K}$ intersections, using the algorithm of Section 3. With some care we can obtain such a partition with at most one 5-cell, which we then split by a diagonal into two subcells. This yields a final collection of quadrilaterals $\{\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_M\}$, which is the output of the algorithm.

4.3 Analysis of the algorithm

The algorithm just described for partitioning a convex quadrilateral is fairly simple. Its analysis, however, is not. For $i = 1, \ldots, M$, let m_i denote the number of lines passing through Q_i, and let K_i denote the number of intersections contained in Q_i. For simplicity we assume that the lines are in general position, that is, no three lines are concurrent, and no line is vertical. We begin by bounding the total number of cells created by the algorithm and the number of lines meeting each cell. As we shall see later, the constants appearing in the bounds for $\max_i m_i$ and $\sum_i m_i$ control the exponent of the logarithmic factor in the bound for the time complexity of the overall partitioning algorithm, therefore we try to obtain as small constants as possible. If we do not worry about constants, the proofs can be simplified a lot. To bound the total number of cells produced by our algorithm, we first estimate the number of cells formed by overlapping the original red, green and blue pseudo edges.

Lemma 4.7 *Let N be the number of cells produced by overlapping the red, green and blue pseudo edges, then*

$$N \leq \frac{2}{\zeta} m + \frac{K}{\zeta^2} - 2. \tag{4.7}$$

Proof: We bound the number of boundary and internal cells separately. The number of boundary cells is obviously equal to the number of endpoints of pseudo edges, namely $2|\mathcal{S}| = 2(u + v + t - 3)$.

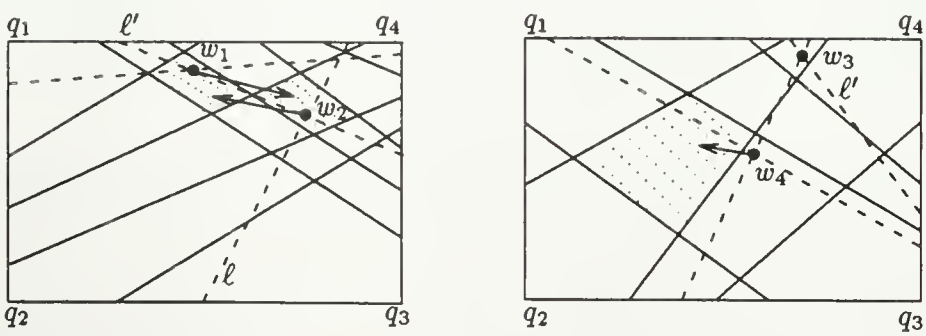


Figure 9: Charging of a red-green intersection point

Next, consider the number of internal cells. To bound the number of red-green and red-blue-green cells, we use the following charging scheme. Let w be the intersection point of a red line ℓ and a green line ℓ'. Suppose that the right endpoint of ℓ lies on $\overline{q_1 q_4}$ in a cell Q_j^r, and that the left endpoint of ℓ' (on $\overline{q_1 q_4}$) lies in Q_i^g. If $i = 1$ and $1 < j < t$, then

we charge w to the red-blue-green cell $\mathcal{F}_{1j}$ (e.g. w_4 in figure 9) , and if $1 < i < u$ and $1 < j < t$, then we charge w to the red-green cell $\mathcal{F}_{ij}$ (e.g. w_2 in figure 9). If $\mathcal{Q}_i^g \cap \mathcal{Q}_j^r$ is a boundary cell, then we do not charge w to any cell (e.g. w_3 in figure 9). Similarly, if the right endpoint of ℓ lies on $\overline{q_3q_4}$, then we charge w to the cell $\mathcal{F}_{ij}$ (e.g. w_1 in figure 9), where $\mathcal{Q}_i^g$ contains the right endpoint of ℓ' (on $\overline{q_3q_4}$) and $\mathcal{Q}_j^r$ contains, as before, the right endpoint of ℓ. Again, no charge is made if $\mathcal{Q}_i^g \cap \mathcal{Q}_j^r$ is a boundary cell.

Clearly, each internal red-green intersection is charged to at most one cell. Moreover, it is easily checked that each internal red-green or red-blue-green cell is charged exactly ζ^2 red-green intersections, namely those between the red lines whose right endpoint lies in $\mathcal{Q}_j^r$ and the green lines whose right or left endpoint (as the case might be) lies in $\mathcal{Q}_i^g$ (by construction, all those intersections points do lie inside $\mathcal{Q}$). Hence there are at most $\frac{K_{rg}}{\zeta^2}$ red-green and red-blue-green cells. Similarly we can prove that there are at most $\frac{K_{rb}}{\zeta^2}$ red-blue cells. Thus, the overall number N of cells after the third step is bounded by

$$\begin{aligned} N &\leq 2(t+u+v) + \frac{K_{rg}}{\zeta^2} + \frac{K_{rb}}{\zeta^2} - 6 \\ &\leq 2\left(\left\lceil\frac{m_r}{\zeta}\right\rceil + \left\lceil\frac{m_g}{\zeta}\right\rceil + \left\lceil\frac{m_b}{\zeta}\right\rceil\right) + \frac{1}{\zeta^2}(K_{rg}+K_{rb}) - 6 \\ &\leq \frac{2}{\zeta}(m+2) + \frac{1}{\zeta^2}(K_{rg}+K_{rb}) - 6 \\ &\leq \frac{2}{\zeta}m + \frac{K}{\zeta^2} - 2 \end{aligned}$$

□

Lemma 4.8 *Let M be the total number of cells produced by the above algorithm. Then*

$$M \leq \frac{3}{\zeta}m + \frac{2K}{\zeta^2}.$$

Proof: It is easy to see that for any two adjacent boundary cells, one of them has at most four edges, therefore there are at most $|\mathcal{S}|$ cells with five or more edges (see figure 10). Moreover there is at most one cell with seven or eight edges. Since each 5 or 6-cell is partitioned into two convex quadrilaterals and the 7 or 8-cell into three convex quadrilaterals, the fourth step creates at most $|\mathcal{S}|+1$ convex quadrilaterals, which in conjunction with (4.7) implies that after the fourth step the number of cells is at most

$$\frac{2m}{\zeta} + \frac{K}{\zeta^2} - 2 + (t+u+v-3) \leq \frac{3m}{\zeta} + \frac{K}{\zeta^2} - 5.$$

Finally, the fifth step adds at most $\lceil\frac{K_\alpha}{\zeta\sqrt{K}}\rceil$ new cells at each quadrilateral containing $K_\alpha > \zeta\sqrt{K}$ intersection points. Since there can be at most $\frac{K}{\zeta\sqrt{K}}$ such quadrilaterals, the

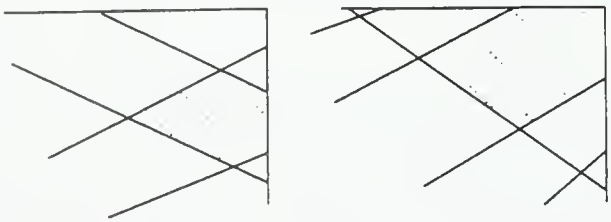


Figure 10: No two adjacent cells have more than four edges

total number of new cells is at most

$$\sum_{\alpha}\left(\frac{K_\alpha}{\zeta\sqrt{K}}+1\right) \le \frac{2\sqrt{K}}{\zeta} \le \frac{K}{\zeta^2},$$

because we apply this step only when $K > 6\zeta^2$. Thus

$$M \le \frac{3m}{\zeta}+\frac{2K}{\zeta^2}.$$

□

Next, we bound the maximum number of lines passing through any cell and the total number of line-cell crossings.

Lemma 4.9 *For each $1 \le i \le M$, the number m_i of lines passing through the cell $\mathcal{Q}_i$ satisfies*

$$m_i \le \max\{2\sqrt{2}\sqrt{K}+4\zeta,\ 2\sqrt{K}+6\zeta\}. \tag{4.8}$$

Proof: Note that a cell $\mathcal{Q}_i$ produced by our algorithm is a subset of some cell $\mathcal{F}_\alpha \in \mathbf{F}$ (the cells obtained by overlapping red, blue and green pseudo edges). Therefore it suffices to bound the number of lines passing through a cell in $\mathbf{F}$. First, let us consider a red-green cell $\mathcal{F}_{ij} = \mathcal{Q}_i^g \cap \mathcal{Q}_j^r$. By Lemma 4.5, $\mathcal{Q}_i^g$ meets at most $2\sqrt{K_{gg}}+2\zeta$ green lines and $\mathcal{Q}_j^r$ meets at most $2\sqrt{K_{rr}}+2\zeta$ red lines, and since no blue line passes through a red-green cell, there are at most $2(\sqrt{K_{rr}}+\sqrt{K_{gg}}+2\zeta)$ lines pass through $\mathcal{F}_{ij}$. Similarly, we can show that a red-blue cell meets at most $2(\sqrt{K_{rr}}+\sqrt{K_{bb}}+2\zeta)$ lines.

Finally, let $\mathcal{F}_{1i}$ be a red-blue-green cell, then $\mathcal{F}_{1i}$ is the intersection of $\mathcal{Q}_1^g$, $\mathcal{Q}_1^b$ and $\mathcal{Q}_i^r$. Since $\mathcal{Q}_1^g$ (resp. $\mathcal{Q}_1^b$) meets at most 2ζ green (resp. blue) lines, $\mathcal{F}_{1i}$ has at most $2\sqrt{K_{rr}}+6\zeta$ lines. Now the lemma follows from the fact that $\max\{\sqrt{K_{rr}}+\sqrt{K_{bb}},\ \sqrt{K_{rr}}+\sqrt{K_{gg}}\} \le \sqrt{2K}$.

□

For $x, y \in \{r, b, g\}$, let δ_i^{xy} denote the number of pseudo edges of color x intersected by the line ℓ_i of color y. It follows from Lemma 4.2 that

$$\sum_{i=1}^{m_x}\delta_i^{xx} \le \frac{2K_{xx}}{\zeta}+m_x.$$

In the next two lemmas we bound the "mixed" sums $\sum_{i=1}^{m_g} \delta_i^{rg}$, $\sum_{i=1}^{m_b} \delta_i^{rb}$, $\sum_{i=1}^{m_r} \delta_i^{gr}$ and $\sum_{i=1}^{m_r} \delta_i^{br}$.

Lemma 4.10 $\displaystyle\sum_{i=1}^{m_g} \delta_i^{rg} \le \frac{K_{rg}}{\zeta} + m_g$ and $\displaystyle\sum_{i=1}^{m_b} \delta_i^{rb} \le \frac{K_{rb}}{\zeta} + m_b.$

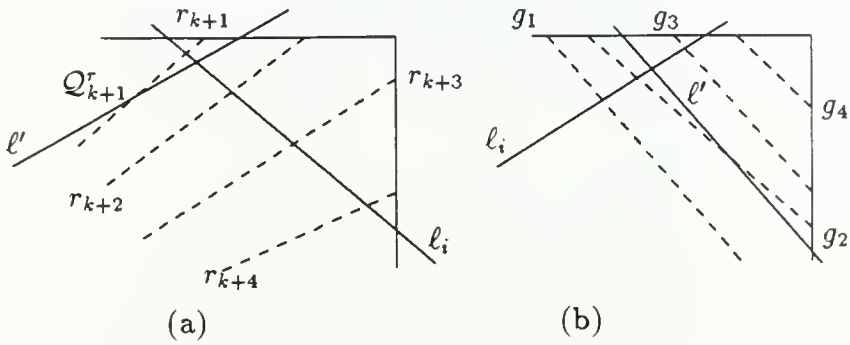


Figure 11: Illustration for Lemma 4.10 and Lemma 4.11

Proof: We will only prove the first part of the lemma; the second part can be proved in a symmetric way. Let ℓ_i be a green line intersecting δ_i^{rg} red pseudo edges, $r_{k+1}, \ldots, r_{k+\delta_i^{rg}}$, and let ℓ' be a red line having its right endpoint in Q_l^r, for $k+1 < l < k+\delta_i^{rg}$ (see figure 11a). It is easily checked that ℓ' intersects ℓ_i inside Q. Since each Q_l^r contains the right endpoints of ζ lines, ℓ_i intersects at least $(\delta_i^{rg}-1)\zeta$ lines inside Q. But there are at most K_{rg} red-green intersection points, therefore summing over all green lines, we obtain

$$\sum_{i=1}^{m_g}(\delta_i^{rg}-1)\zeta \le K_{rg} \quad \text{or} \quad \sum_{i=1}^{m_g}\delta_i^{rg} \le \frac{K_{rg}}{\zeta} + m_g$$

□

Lemma 4.11 $\displaystyle\sum_{i=1}^{m_r} \delta_i^{gr} \le \frac{K_{rg}}{\zeta}$ and $\displaystyle\sum_{i=1}^{m_r} \delta_i^{br} \le \frac{K_{rb}}{\zeta}.$

Proof: Let ℓ_i be a red line intersecting δ_i^{gr} green pseudo edges, $g_1, \ldots, g_{\delta_i^{gr}}$. Suppose that the right endpoint of ℓ_i lies on $\overline{q_1q_4}$. Let ℓ' be a green line whose left endpoint (on $\overline{q_1q_4}$) lies in Q_l^g, for $l \le \ell_i^{gr}$ (see figure 11b). It is easily seen that ℓ' intersects ℓ_i inside Q. Similarly, if the right endpoint of ℓ_i is on $\overline{q_3q_4}$, then ℓ_i intersects any green line whose right endpoint (on $\overline{q_3q_4}$) lies in Q_l^g, for $l \le \ell_i^{gr}$. Since each Q_l^g contains ζ left endpoints, and also ζ right endpoints, of green lines, ℓ_i intersects at least $\delta_i^{gr}\cdot\zeta$ green lines inside Q. Summing over all red lines, we obtain

$$\sum_{i=1}^{m_r}\delta_i^{gr}\cdot\zeta \le K_{rg} \quad \text{or} \quad \sum_{i=1}^{m_r}\delta_i^{gr} \le \frac{K_{rg}}{\zeta}.$$

Similarly, we can prove that $\sum_{i=1}^{m_r}\delta_i^{br} \le \frac{K_{rb}}{\zeta}$.

□

Lemma 4.12 *Let m'_i denote the number of lines passing through the ith cell of* **F**, *then*

$$\sum_{i=1}^{N} m'_i \leq \frac{2}{\zeta}K + \frac{5}{2}m \tag{4.9}$$

Proof: Let η_j^x denote the number of cells crossed by a line $\ell_j \in \mathcal{L}_x$, where $x \in \{r, b, g\}$, Each cell $\mathcal{F}$ meeting ℓ_j is charged to the leftmost point $w_{\mathcal{F}}$ of $\partial\mathcal{F} \cap \ell_j$, therefore for all cells crossed by ℓ_j, except the leftmost one, $w_{\mathcal{F}}$ is an intersection point of a pseudo edge and ℓ_j (see figure 12). For red lines it follows from Lemma 4.2 and Lemma 4.11 that

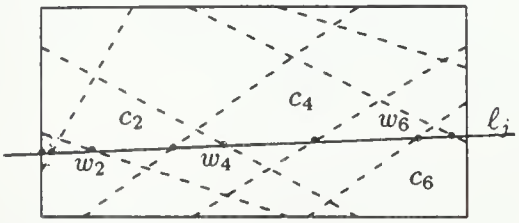


Figure 12: Line ℓ_j and the cells in **F**; cell c_i is charged to w_i

$$\begin{aligned}\sum_{i=1}^{m_r} \eta_i^r &= \sum_{i=1}^{m_r} (\delta_i^{rr} + \delta_i^{gr} + \delta_i^{br} + 1) \\ &\leq 2\frac{K_{rr}}{\zeta} + \frac{K_{rg}}{\zeta} + \frac{K_{rb}}{\zeta} + 2m_r \end{aligned} \tag{4.10}$$

Similarly, using Lemma 4.2 and Lemma 4.10 we can show that

$$\begin{aligned}\sum_{i=1}^{m_b} \eta_i^b &= \sum_{i=1}^{m_b} (\delta_i^{bb} + \delta_i^{rb} + \delta_i^{gb} + 1) \\ &\leq 2\frac{K_{bb}}{\zeta} + \frac{K_{rb}}{\zeta} + 3m_b, \end{aligned} \tag{4.11}$$

$$\text{and} \quad \sum_{i=1}^{m_g} \eta_i^g \leq 2\frac{K_{gg}}{\zeta} + \frac{K_{rg}}{\zeta} + 3m_g \tag{4.12}$$

Since $\sum_{i=1}^{N} m'_i = \sum_{\substack{x \in \\ \{r,b,g\}}} \sum_{j=1}^{m_x} \eta_j^x$, summing (4.10), (4.11) and (4.12), we obtain

$$\begin{aligned}\sum_{i=1}^{N} m'_i &\leq \frac{2}{\zeta}(K_{rr} + K_{bb} + K_{gg} + K_{rg} + K_{rb}) + 3(m_g + m_b) + 2m_r \\ &\leq \frac{2}{\zeta}K + \frac{5}{2}m \qquad (\text{ because } m_r \geq \tfrac{m}{2}; \text{ see Section 2})\end{aligned}$$

□

Next, we bound $\sum_{i=1}^{M} m_i$, that is the total number of line-cell crossings, for the final cells produced by the algorithm.

Lemma 4.13 *Let m_i denote the number of lines in cell Q_i; then*

$$\sum_{i=1}^{M} m_i \leq (4+4\sqrt{2})\frac{K}{\zeta} + (5+4\sqrt{2})m.$$

Proof: Let m_α denote the number of lines meeting the αth cell after the fourth step. First consider the case when **F** has a cell $\mathcal{F}_\alpha$ with seven or eight edges, say eight edges (see figure 8). Let r_{i-1} and r_i denote the red pseudo lines bounding $\mathcal{F}_\alpha$. In this case the fourth step adds only two pseudo edges, γ_L and γ_R. Since γ_L does not intersect green lines and intersects only those red lines whose left endpoints lie in Q_i^r, it adds at most $\zeta + m_b$ line-cell crossings. Similarly γ_R adds at most $\zeta + m_g$ line-cell crossings. Moreover, if **F** has a 7-cell, then it can also have one 5-cell. Thus by Lemma 4.12,

$$\sum_\alpha m_\alpha \leq \frac{2}{\zeta}K + \frac{5}{2}m + (2\zeta + m_g + m_b) + m$$

Note that in this case the first three steps create at least three pseudo edges, which implies that $m > 3\zeta$. Since $m_g + m_b \leq \frac{m}{2}$, we have

$$\sum_\alpha m_\alpha \leq \frac{2}{\zeta}K + 5m.$$

On the other hand if all cells after the third step have at most six edges, then every cell is partitioned into at most two cells. Thus by Lemma 4.12, the total number of line-cell crossing after the fourth step is

$$\sum_\alpha m_\alpha \leq \frac{4}{\zeta}K + 5m.$$

The final step creates at most $\frac{2\sqrt{K}}{\zeta}$ new cells. Moreover, in that step a cell $\mathcal{F}_\alpha$ is partitioned only when $K > 6\zeta^2$, in which case $m_\alpha < 2\sqrt{2}\sqrt{K} + 4\zeta$ (cf. Lemma 4.9). Hence, we obtain

$$\begin{aligned}\sum_{i=1}^{M} m_i &\leq \frac{4}{\zeta}K + 5m + \frac{2\sqrt{K}}{\zeta}(2\sqrt{2}\sqrt{K} + 4\zeta) \\ &= (4+4\sqrt{2})\frac{K}{\zeta} + 8\sqrt{K} + 5m \\ &\leq (4+4\sqrt{2})\frac{K}{\zeta} + (5+4\sqrt{2})m\end{aligned}$$

because $K \leq \frac{m^2}{2}$

□

Finally, we analyze the running time of the algorithm.

Lemma 4.14 *It requires $O\left((m+\frac{K}{\zeta})\log m\right)$ time to compute all subquadrilaterals $\mathcal{Q}_1, \ldots, \mathcal{Q}_M$, to determine the lines passing through each $\mathcal{Q}_i$, and to count the number of intersection points in each $\mathcal{Q}_i$.*

Proof: It follows from Lemma 4.5 that $\mathbf{Q}^r$, $\mathbf{Q}^g$ and $\mathbf{Q}^b$ can be computed in time $O((m_r + m_b + m_g)\log m) = O(m \log m)$. Once we have computed all pseudo edges, the set $\mathbf{F}$ of cells, obtained by overlaying these pseudo edges, can be computed, for example, by sweeping a vertical line through the quadrilateral $\mathcal{Q}$. The time spent in the sweep is $O((|\mathcal{S}|+|\mathbf{F}|)\log \mathcal{S})$ $= O\left((\frac{m}{\zeta}+\frac{K}{\zeta^2})\log m\right)$, because it encounters at most $O(|\mathbf{F}|)$ intersection points. Since step 4 simply scans all boundary cells and splits those having more than four edges, it can be easily done in $O(|\mathcal{S}|)$ time.

As for the final step, let m_α (resp. K_α) be the number of lines meeting (resp. intersection points contained in) the αth cell produced by the first four steps. The total time spent in computing the lines passing through each cell $\mathcal{F}_\alpha$ is at most $O(m \log m + \sum_\alpha m_\alpha \log m) = O((m+\frac{K}{\zeta})\log m)$ (cf. Lemma 4.13), and it follows from Lemma 3.1 that the quantities K_α, for all cells $\mathcal{F}_\alpha$, can be calculated in time $O((m+\frac{K}{\zeta})\log m)$. Moreover, step 5 of the algorithm partitions the αth cell into at most $\lceil \frac{K_\alpha}{\zeta\sqrt{K}} \rceil$ convex quadrilaterals, which is done applying the algorithm of Section 3 at most $\lceil \frac{K_\alpha}{\zeta\sqrt{K}} \rceil - 1$ times, each application taking $O(m_\alpha \log m_\alpha)$ time (cf. Theorem 3.3). Therefore, the total time spent in the final step is bounded by

$$
\begin{aligned}
& O\Big(\sum_\alpha \Big(\Big\lceil \frac{K_\alpha}{\zeta\sqrt{K}} \Big\rceil - 1\Big) m_\alpha \log m_\alpha\Big) \\
= \; & O\Big(\frac{1}{\zeta\sqrt{K}} \sum_\alpha K_\alpha \cdot m_\alpha \log m_\alpha\Big) \\
= \; & O\Big(\frac{\sqrt{K}+\zeta}{\zeta\sqrt{K}} \log m \Big(\sum_\alpha K_\alpha\Big)\Big) \\
& \text{(because } m_\alpha \le 2\sqrt{2}\sqrt{K} + 4\zeta) \\
= \; & O\Big((\frac{K}{\zeta} + \sqrt{K})\log m\Big) \\
= \; & O\Big((\frac{K}{\zeta} + m)\log m\Big) \\
& \text{(because } K \le \frac{m^2}{2}).
\end{aligned}
$$

Thus, the overall running time of our algorithm is $O\left((m+\frac{K}{\zeta})\log m\right)$.

□

Hence, we can conclude that

Theorem 4.15 *We can partition the convex quadrilateral $\mathcal{Q}$ into $M \leq \frac{3m}{\zeta} + \frac{2K}{\zeta^2}$ convex subquadrilaterals $\mathcal{Q}_1, \ldots, \mathcal{Q}_M$, with the property that each $\mathcal{Q}_i$ is crossed by m_i lines and contains K_i intersection points, so that the following conditions are satisfied:*

$$\begin{array}{lll} (i) & \sum_{i=1}^{M} m_i \leq A_1 m + A_2 \frac{K}{\zeta}; & \max m_i \leq A_3\sqrt{K} + A_4\zeta \\ (ii) & \sum_{i=1}^{M} K_i = K; & \max K_i \leq \zeta\sqrt{K} \end{array}$$

where $A_1 = (5 + 4\sqrt{2})$, $A_2 = (4 + 4\sqrt{2})$, $A_3 = 2\sqrt{2}$ and $A_4 = 6$.

□

5 Partitioning the plane into quadrilaterals

In this section we obtain the main result of this paper. Let $\mathcal{L} = \{\ell_1, \ell_2, \ldots, \ell_n\}$ be a set of n lines in the plane, and $\mathbf{R}$ an "enclosing rectangle" of $\mathcal{L}$, i.e. one that contains all $\binom{n}{2}$ intersection points of $\mathcal{L}$. For a given integer $r > 0$, we want to partition $\mathbf{R}$ into $O(r^2 \log^\beta r)$ convex quadrilaterals, for some constant $\beta > 0$ to be determined later, so that each of them is crossed by at most $\frac{n}{r}$ lines.

The idea is to use the algorithm of the previous section recursively. Fix $\zeta = \frac{n}{\gamma r}$, where γ is a constant to be chosen later (for simplicity let us assume that n is a multiple of γr). At each step of the algorithm we have a convex quadrilateral $\mathcal{Q}$, which meets m of the given lines and contains K of their intersection points. Initially $\mathcal{Q} = \mathbf{R}$, $m = n$ and $K = \binom{n}{2}$ (assuming that the lines are in general position). At each recursive step, the algorithm proceeds as follows: If $m \leq \frac{n}{r}$, there is nothing to do, so we stop; otherwise partition $\mathcal{Q}$ into M convex quadrilaterals using the algorithm of the previous section (for the initial fixed value of ζ).

Let m_i denote the number of lines meeting the convex quadrilateral $\mathcal{Q}_i$ and let K_i denote the number of intersection points of $\mathcal{L}$ contained in $\mathcal{Q}_i$. Let $\mathcal{C}(m, K)$ denote the maximum number of cells into which such a $\mathcal{Q}$ will be partitioned by all subsequent recursive applications of the algorithm, and let $\mathcal{T}(m, K)$ denote the maximum time required for such a partitioning. It follows from Theorem 4.15 that

$$\mathcal{C}(m, K) \leq \begin{cases} \sum_{i=1}^{M} \mathcal{C}(m_i, K_i) & \text{if } m > \frac{n}{r} \\ 1 & \text{otherwise} \end{cases} \tag{5.1}$$

$$\mathcal{T}(m, k) \leq \begin{cases} \sum_{i=1}^{M} \mathcal{T}(m_i, K_i) + D\left(m + \frac{K}{\zeta}\right) \log m & \text{if } m > \frac{n}{r} \\ D & \text{otherwise} \end{cases} \tag{5.2}$$

where D is some constant > 0 and M, m_i and K_i satisfy the bounds given in Theorem 4.15.

Next we bound the values of $\mathcal{C}(m,K)$ and $\mathcal{T}(m,K)$ using (5.1) and (5.2). In what follows by $\mathrm{Log}x$ we mean $\max\{\log x, 1\}$, and similarly, later on, by $\mathrm{LogLog}x$ we mean $\max\{\log\log x, \log\log\sqrt{6}\}$.

Lemma 5.1 *There exists a constant $E > 0$ such that*

$$\mathcal{C}(m,K) \le E\left(\frac{m}{\zeta} + \frac{K}{\zeta^2}\right)\mathrm{Log}^{\beta}\frac{K}{\zeta^2}, \tag{5.3}$$

where $\beta = \max\{\log A_1, \log(1+A_2)\} < 3.33$.

Proof: We prove the above inequality by induction on K. If we choose $\gamma \ge 4(\sqrt{3}+1)$, then for $K \le 6\zeta^2$, after applying the algorithm once, we have

$$\begin{aligned}\max m_i &\le \max\{2\sqrt{2}\sqrt{K} + 4\zeta,\ 2\sqrt{K} + 6\zeta\}\\ &= \max\{(4\sqrt{3}+4)\zeta,\ (2\sqrt{6}+6)\zeta\}\\ &= 4(\sqrt{3}+1)\zeta \quad \le \quad \frac{n}{r}\end{aligned}$$

and therefore, the algorithm stops. By Theorem 4.15 this step partitions $\mathcal{Q}$ into at most $3\left(\frac{m}{\zeta} + \frac{K}{\zeta^2}\right)$ cells. Thus, if we choose $E > 3$, then for $K \le 6\zeta^2$, (5.3) holds trivially.

For $K > 6\zeta^2$, suppose inductively that (5.3) holds for all $K' < K$. Since $K_i \le \zeta\sqrt{K}$ and $K > 6\zeta^2$, $K_i \le \sqrt{\frac{K}{6}} \cdot \sqrt{K} = \frac{K}{\sqrt{6}} < K$. Therefore by induction hypothesis, (5.3) holds for all $\mathcal{Q}_i$, so (5.1) implies

$$\mathcal{C}(m,K) \quad \le \quad \sum_{i=1}^{M} E\left(\frac{m_i}{\zeta} + \frac{K_i}{\zeta^2}\right)\mathrm{Log}^{\beta}\frac{K_i}{\zeta^2} \tag{5.4}$$

Since $\forall i,\ K_i \le \zeta\sqrt{K}$, we have

$$\mathrm{Log}\frac{K_i}{\zeta^2} \le \mathrm{Log}\frac{\zeta\sqrt{K}}{\zeta^2} = \max\{\log\sqrt{\frac{K}{\zeta^2}}, 1\}.$$

$K \ge 6\zeta^2$ implies that $\log\sqrt{\frac{K}{\zeta^2}} > 1$, therefore (5.4) can be written as

$$\begin{aligned}\mathcal{C}(m,K) &\le \sum_{i=1}^{M} E\left(\frac{m_i}{\zeta} + \frac{K_i}{\zeta^2}\right)\log^{\beta}\sqrt{\frac{K}{\zeta^2}}\\ &= E\left(\frac{1}{\zeta}\sum_{i=1}^{M} m_i + \frac{1}{\zeta^2}\sum_{i=1}^{M} K_i\right)\cdot\frac{1}{2^{\beta}}\log^{\beta}\frac{K}{\zeta^2}\\ &\le \frac{E}{2^{\beta}}\left(A_1\frac{m}{\zeta} + (A_2+1)\frac{K}{\zeta^2}\right)\log^{\beta}\frac{K}{\zeta^2}\end{aligned}$$

(By Theorem 4.15)

Let $\beta = \max\{\log A_1, \log(1 + A_2)\}$, then

$$\frac{1}{2^\beta}\left(A_1\frac{m}{\zeta} + (A_2+1)\frac{K}{\zeta^2}\right) \le \left(\frac{m}{\zeta} + \frac{K}{\zeta^2}\right)$$

Therefore

$$\begin{aligned} \mathcal{C}(m,K) &\le E\left(\frac{m}{\zeta} + \frac{K}{\zeta^2}\right)\log^\beta\frac{K}{\zeta^2} \\ &\le E\left(\frac{m}{\zeta} + \frac{K}{\zeta^2}\right)\mathrm{Log}^\beta\frac{K}{\zeta^2}. \end{aligned}$$

□

Remark 5.2: The actual value of β depends on the constants appearing in Theorem 4.15(i). We believe that these constants are not optimal, so β is likely to be smaller than $\log_2(5 + 4\sqrt{2}) = 3.33$.

Next, we bound the running time $\mathcal{T}(m,K)$ of the algorithm.

Lemma 5.3 *There exists a constant $F > 0$ such that*

$$\mathcal{T}(m,K) \le F\left(m + \frac{K}{\zeta}\right)\log m \cdot \mathrm{Log}^\beta\frac{K}{\zeta^2} \cdot \mathrm{LogLog}\frac{K}{\zeta^2}. \tag{5.5}$$

Proof: Again we prove the inequality by induction on K. In Lemma 5.1 we showed that, for $K \le 6\zeta^2$, the algorithm stops after one step of recursion. By Theorem 4.15, this step requires $O((m + \frac{K}{\zeta})\log m)$ time, therefore if we choose F sufficiently large, then (5.5) holds trivially.

For $K > 6\zeta^2$, suppose inductively that (5.5) holds for all $K' < K$. Since $K_i < K$ (cf. Lemma 5.1), by induction hypothesis, (5.5) holds for all $\mathcal{Q}_i$, so (5.2) implies

$$\begin{aligned} \mathcal{T}(m,K) \le\ & \sum_{i=1}^{M} F\left(m_i + \frac{K_i}{\zeta}\right)\log m_i \cdot \mathrm{Log}^\beta\frac{K_i}{\zeta^2} \cdot \mathrm{LogLog}\frac{K_i}{\zeta^2} + \\ & D\left(m + \frac{K}{\zeta}\right)\log m \end{aligned} \tag{5.6}$$

Note that, for all i, $K_i \le \zeta\sqrt{K}$, therefore

$$\mathrm{Log}^\beta\frac{K_i}{\zeta^2}\mathrm{LogLog}\frac{K_i}{\zeta^2} \le \max\{\log^\beta\sqrt{\frac{K}{\zeta^2}}, 1\} \cdot \max\{\log\log\sqrt{\frac{K}{\zeta^2}}, \log\log\sqrt{6}\}.$$

But $K \ge 6\zeta^2$, therefore it is easily seen that

$$\mathrm{Log}^\beta\sqrt{\frac{K}{\zeta^2}}\mathrm{LogLog}\sqrt{\frac{K}{\zeta^2}} = \log^\beta\sqrt{\frac{K}{\zeta^2}}\log\log\sqrt{\frac{K}{\zeta^2}}. \tag{5.7}$$

Substituting it in (5.6), we obtain

$$\begin{aligned}
\mathcal{T}(m,K) &\leq \sum_{i=1}^{M} F\left(m_i + \frac{K_i}{\zeta}\right)\log m \cdot \log^\beta \sqrt{\frac{K}{\zeta^2}} \cdot \log\log\sqrt{\frac{K}{\zeta^2}} + D\left(m + \frac{K}{\zeta}\right)\log m \\
&\leq \left(\sum_{i=1}^{M} m_i + \frac{1}{\zeta}\sum_{i=1}^{M} K_i\right)\frac{F}{2^\beta}\cdot \log^\beta \frac{K}{\zeta^2}\left(\log\log\frac{K}{\zeta^2} - 1\right)\log m + \\
&\quad D\left(m + \frac{K}{\zeta}\right)\log m \\
&\leq \left(A_1 m + (A_2+1)\frac{K}{\zeta}\right)\frac{F}{2^\beta}\cdot \log^\beta \frac{K}{\zeta^2}\left(\log\log\frac{K}{\zeta^2} - 1\right)\log m + \\
&\quad D\left(m + \frac{K}{\zeta}\right)\log m \qquad \text{(by Theorem 4.15)} \\
&\leq \left(m + \frac{K}{\zeta}\right)\left[F\log^\beta\frac{K}{\zeta^2}\cdot\log\log\frac{K}{\zeta^2} + \left(D - F\log^\beta\frac{K}{\zeta^2}\right)\right]\log m
\end{aligned}$$

because $\beta = \max\{\log A_1, \log(A_2+1)\}$. Since $K > 6\zeta^2$, we have $F\log^\beta \frac{K}{\zeta^2} > 2^\beta$, and therefore, by choosing $F = \frac{D}{2^\beta}$, we obtain

$$\begin{aligned}
\mathcal{T}(m,K) &\leq F\left(m + \frac{K}{\zeta}\right)\log m \cdot \log^\beta\frac{K}{\zeta^2}\cdot\log\log\frac{K}{\zeta^2} \\
&\leq F\left(m + \frac{K}{\zeta}\right)\log m \cdot \mathrm{Log}^\beta\frac{K}{\zeta^2}\cdot\mathrm{LogLog}\frac{K}{\zeta^2}.
\end{aligned}$$

□

These lemmas imply that

Theorem 5.4 *Given a set $\mathcal{L}$ of n lines in the plane and a parameter $1 < r < n$, we can partition an enclosing rectangle $\mathbf{R}$ into $O(r^2\log^\beta r)$ convex quadrilaterals in time $O(nr\log n\log^\beta r\log\log r)$, so that each quadrilateral meets at most $\frac{n}{r}$ lines of $\mathcal{L}$.*

Proof: Choose $\zeta = \frac{n}{\gamma r}$, where $\gamma = 4(\sqrt{3}+1)$. At the top level of the recursion $m = n$ and $K = \binom{n}{2} < \frac{n^2}{2}$. Therefore

$$\frac{K}{\zeta^2} < \frac{n^2/2}{n^2/\gamma^2 r^2} = \frac{\gamma^2 r^2}{2}$$

and $\dfrac{m}{\zeta} = \dfrac{n}{n/\gamma r} = \gamma r$. Substituting these values in (5.3) we obtain

$$\begin{aligned}
\mathcal{C}(n, n^2/2) &\leq E\left(\gamma r + \frac{\gamma^2 r^2}{2}\right)\mathrm{Log}^\beta\frac{\gamma^2 r^2}{2} \\
&= O\left(r^2\log^\beta r\right)
\end{aligned}$$

and by substituting $\frac{K}{\zeta} = \frac{n^2/2}{2n/\gamma r} = \frac{\gamma r}{4} n$ in (5.5) we get

$$\begin{aligned} \mathcal{T}(n, n^2) &\leq F\left(n + \frac{\gamma r}{4} n\right) \log n \cdot \mathrm{Log}^{\beta} \frac{\gamma^2 r^2}{2} \mathrm{LogLog} \frac{\gamma^2 r^2}{2} \\ &= O\left(nr \log n \log^{\beta} r \log\log r\right) \end{aligned}$$

□

Remark 5.5: To make the notations easier to follow, we shall henceforth replace the term $\log^{\beta} r \cdot \log\log r$ by $\log^{\omega} r$, for some fixed constant ω slightly larger than β. Since we do not know what the best value for β is, this convention involves no real loss of information.

6 Constructing approximate levels

In this section we describe the second phase of our algorithm that reduces the number of triangles from $O(r^2 \log^{\beta} r)$ to $O(r^2)$, while maintaining the property that each triangle meets at most $O(\frac{n}{r})$ lines of $\mathcal{L}$. As mentioned in the introduction, this second phase is not required in most of the applications. As an intermediate step, the algorithm constructs an $\frac{n}{2r}$-approximate leveling of $\mathcal{A}(\mathcal{L})$ (as defined in Section 2), with $O(r^2 \log^{\beta} r)$ edges in total, from the partition obtained in the first phase of the algorithm. Once an $\frac{n}{2r}$-approximate leveling has been constructed, we proceed in the same way as Matoušek [Maa]. We first describe how to obtain an $\frac{n}{2r}$-approximate leveling.

Let $\overline{\mathcal{Q}}$ denote the planar map induced by the preceding partition of the enclosing rectangle $\mathbf{R}$. We assume that all lines in $\mathcal{L}$ intersect $\partial\mathbf{R}$ at its vertical edges. Let $A = \{a_1, a_2, \ldots, a_n\}$ (resp. $B = \{b_1, b_2, \ldots, b_n\}$) denote the intersection points of the lines in $\mathcal{L}$ and the left (resp. right) vertical edge of $\partial\mathbf{R}$, sorted in decreasing order of their y-coordinates. For $1 \leq i \leq r$, add $a_{in/r}$ and $b_{in/r}$ to the set of vertices of $\overline{\mathcal{Q}}$. We triangulate all faces of $\overline{\mathcal{Q}}$, in time $O(r^2 \log^{\beta} r)$. The triangulated map $\mathcal{G}$ also has $O(r^2 \log^{\beta} r)$ edges. The following observations enable us to compute approximate levels efficiently.

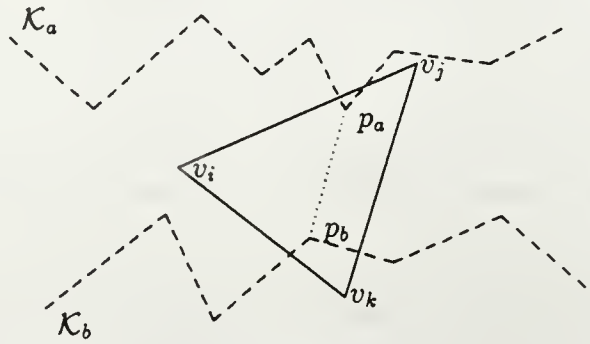


Figure 13: A triangle intersecting both $\mathcal{K}_a$ and $\mathcal{K}_b$.

Lemma 6.1 *Let $\mathcal{K}_a$ (resp. $\mathcal{K}_b$) denote the $k - \frac{n}{2r}$ (resp. $k + \frac{n}{2r}$) level of $\mathcal{A}(\mathcal{L})$. Then the interior of a triangle in $\overline{\mathcal{Q}}$ does not intersect both $\mathcal{K}_a$ and $\mathcal{K}_b$.*

Proof: Suppose to the contrary that there is such a triangle $\Delta = v_i v_j v_k$ that crosses $\mathcal{K}_a$ and $\mathcal{K}_b$. Let p_a (resp. p_b) denote a point in $\mathcal{K}_a \cap \Delta$ (resp. $\mathcal{K}_b \cap \Delta$) (see figure 13). The segment $\overline{p_a p_b}$ lies entirely in Δ, but by Lemma 2.1 it intersects at least $\frac{n}{r} + 1$ lines of $\mathcal{L}$, which contradicts the property that no triangle in $\mathcal{G}$ intersects more than $\frac{n}{r}$ lines of $\mathcal{L}$.

□

Lemma 6.2 *Let $\mathcal{K}_a$ (resp. $\mathcal{K}_b$) denote the $k - \frac{n}{2r}$ (resp. $k + \frac{n}{2r}$) level of $\mathcal{A}(\mathcal{L})$. Let v_0 be a vertex of $\mathcal{G}$ on the left vertical edge of $\mathbf{R}$, whose level is between $k - \frac{n}{2r}$ and $k + \frac{n}{2r}$. There exists a path Π in $\mathcal{G}$ from v_0 to the right vertical edge of $\mathbf{R}$ which lies between $\mathcal{K}_a$ and $\mathcal{K}_b$. Moreover Π can be converted into a monotone path without increasing its number of edges, such that the new path also lies between $\mathcal{K}_a$ and $\mathcal{K}_b$.*

Proof: Let $\mathbf{\Delta} = \Delta_1, \Delta_2, \ldots, \Delta_t$ denote the sequence of triangles visited if we follow $\mathcal{K}_b$ from left to right (if a portion of $\mathcal{K}_b$ lies on an edge of a triangle, then we pick up the triangle lying below that edge). If a triangle appears more than once in $\mathbf{\Delta}$, then retain only its first occurrence. It is easily seen that $\mathbf{\Delta}$ forms a connected region from the left vertical edge to the right vertical edge of $\partial\mathbf{R}$, and its boundary is formed by the edges of $\mathcal{G}$ (see figure 14); we call it a *corridor*. Let Π denote the top portion of $\partial\mathbf{\Delta}$. Π is a connected polygonal path from the left vertical edge of $\partial\mathbf{R}$ to its right vertical edge, formed by the edges of $\mathcal{G}$. Obviously Π does not intersect $\mathcal{K}_b$. Since each triangle of $\mathbf{\Delta}$ intersects $\mathcal{K}_b$, Π cannot intersect $\mathcal{K}_a$ (cf. Lemma 6.1), thus Π lies between $\mathcal{K}_a$ and $\mathcal{K}_b$.

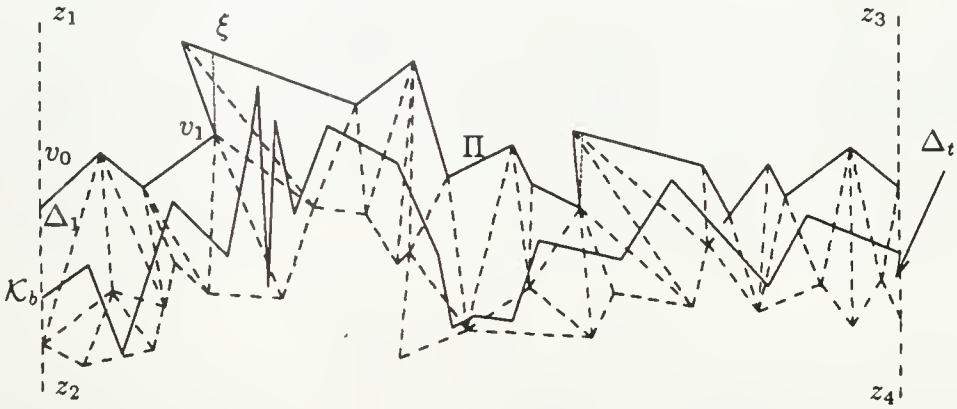


Figure 14: Level $\mathcal{K}_b$ contained in the corridor $\mathbf{\Delta}$; Π is the upper boundary of $\mathbf{\Delta}$.

To prove the second part, let Γ denote any polygonal path lying between $\mathcal{K}_a$ and $\mathcal{K}_b$. Let v_1 be the first vertex on Γ at which Γ turns backwards. Let ξ denote the first point on Γ after v_1 (along Γ), which has the same x-coordinate as v_1. Since v_1 lies above $\mathcal{K}_b$, and $\mathcal{K}_b$ is a x-monotone path, the vertical segment $\overline{v_1\xi}$ does not cross $\mathcal{K}_b$. The same argument

implies that $\overline{v_1\xi}$ does not intersect $\mathcal{K}_a$, and therefore the new path also lies between $\mathcal{K}_a$ and $\mathcal{K}_b$. Whenever we add a new edge, we remove at least one edge of Γ, which proves that shortcuting Γ does not increase its number of edges.

□

Since the path Π lies between the $k-\frac{n}{2r}$ and $k+\frac{n}{2r}$ levels of $\mathcal{A}(\mathcal{L})$, it is an $\frac{n}{2r}$-approximate k-level of $\mathcal{A}(\mathcal{L})$. Therefore $\mathcal{G}$ contains an $\frac{n}{2r}$-approximate $\frac{in}{r}$-level, for each $i \leq r$. Let $\mathcal{G}_k$ denote the subgraph of $\mathcal{G}$ consisting of all edges that fully lie between the levels $k-\frac{n}{2r}$ and $k+\frac{n}{2r}$. To obtain $\mathcal{G}_k$, we need to compute the level of each vertex of $\mathcal{G}$ and the highest and the lowest levels crossed by each edge of $\mathcal{G}$. Let $\psi(v)$ denote the level of a vertex $v \in \mathcal{G}$. Let $\overline{v_iv_j}$ denote an edge of $\mathcal{G}$ with $x(v_i) < x(v_j)$. If there are h lines of $\mathcal{L}$ intersecting $\overline{v_iv_j}$ and k of these lines lie above v_i, then

$$\psi(v_j) \;=\; \psi(v_i) - k + (h-k) \;=\; \psi(v_i) - 2k + h \tag{6.1}$$

For each vertex of $\mathcal{G}$ lying on the left vertical edge of $\partial\mathbf{R}$, we can compute its level by counting the number of lines of $\mathcal{L}$ lying above it. The partitioning algorithm produces the subset of lines of $\mathcal{L}$ that crosses each edge of $\mathcal{G}$, so it is trivial to count how many of them lie above the left endpoint of e. The levels of all vertices of $\mathcal{G}$ are now easy to determine by propagating levels from left to right along the edges of $\mathcal{G}$, using (6.1). As for determining the levels crossed by an edge e of $\mathcal{G}$, we sort the lines intersecting e along the edge. Once we know the level of the endpoints of e, we can easily compute the levels crossed by e. Next we partition $\mathcal{G}$ into $\mathcal{G}_1, \ldots, \mathcal{G}_r$, and find a path from $a_{in/r}$ to $b_{in/r}$ in $\mathcal{G}_i$ using a depth first search from $a_{in/r}$, for $i \leq r$. Finally, if any of the resulting paths is not x-monotone, we make it monotone by shortcuting all edges that turn backwards. Since the edges in $\mathcal{G}_i$, for $i \leq r$, are pairwise disjoint, there are at most $O(r^2 \log^\beta r)$ edges in the resulting $\frac{n}{2r}$-approximate leveling of $\mathcal{A}(\mathcal{L})$.

The correctness of the algorithm follows immediately from the above discussion, so we only have to analyze the running time of the algorithm.

Lemma 6.3 *Given a set $\mathcal{L}$ of n lines, we can construct, in time $O(nr \log n \log^\omega r)$, an $\frac{n}{2r}$-approximate leveling of $\mathcal{A}(\mathcal{L})$ having only $O(r^2 \log^\beta r)$ edges in total.*

Proof: By Theorem 5.4, the planar graph $\mathcal{G}$ can be constructed in time $O(nr \log n \cdot \log^\omega r)$ and it has only $O(r^2 \log^\beta r)$ triangles. It follows from the above discussion that it takes $O(nr \log n \log^\omega r)$ time to compute the level of each vertex and the levels crossed by each edge of $\mathcal{G}$. Therefore, we can obtain $\mathcal{G}_i$, for $i \leq r$, in $O(nr \log n \log^\omega r)$ time. The depth first search takes only $O(r^2 \log^\beta r)$ time, and it takes the same amount of time to convert the computed paths into monotone paths. Hence, the lemma follows.

□

Remark 6.4: Matoušek [Maa] also constructs an approximate leveling as an intermediate step, but in a direct and much simpler (albeit inefficient) manner. His algorithm works roughly as follows.

Partition $\mathbf{R}$ into r^2 vertical slabs each containing at most $O(\frac{n}{r^2})$ intersection points of $\mathcal{L}$. Let $V_0, \ldots, V_{r^2}$ denote the vertical edges of these slabs, and let $a_{i,1}, \ldots, a_{in}$ denote the intersection points of V_i and $\mathcal{L}$ sorted in decreasing order of their y-coordinates. For each $j \leq \frac{r}{2}$ and $0 \leq i < r^2$ connect $a_{i,2jn/r}$ to $a_{i+1,2jn/r}$. These polygonal paths are shown to form an $\frac{n}{r}$-approximate leveling of $\mathcal{A}(\mathcal{L})$.

The problem with this approach is that the approximate leveling has $O(r^3)$ edges in total, and the time needed to obtain it is $O(nr^2 \log^2 r)$, which is substantially dominated by the partitioning of $\mathbf{R}$ into vertical slabs. Partitioning $\mathbf{R}$ into r^2 slabs is done to ensure that no segment $\overline{a_{i,2jn/r}a_{i+1,2jn/r}}$ crosses too many levels. Using our improved partitioning technique we are able to obtain an $\frac{n}{2r}$-approximate partitioning that has almost an order of magnitude fewer edges (in terms of r), and the running time of our algorithm is also about an order of magnitude faster. If r is small, e.g. $O(1)$, then Matoušek algorithm is better (it runs in optimal linear time), but for large values of r it becomes very inefficient. As we will see in [Aga], in most of the applications it is desirable to use a large value of r.

After computing a $\frac{n}{2r}$-approximate leveling of $\mathcal{A}(\mathcal{L})$, we apply the same technique of Matoušek to partition R into $O(r^2)$ triangles. Let $K_1, K_2, \ldots, K_r$ denote the set of $\frac{n}{2r}$-approximate levels of $\mathcal{A}(\mathcal{L})$. Using Lemma 2.2 and Lemma 2.3, Matoušek proved that

Lemma 6.5 ([Maa]) *There exists a polygonal path Π_i between K_{3i-1} and K_{3i+1} such that $\sum_{i=1}^{r/3} |\Pi_i| = O(r^2)$.*

□

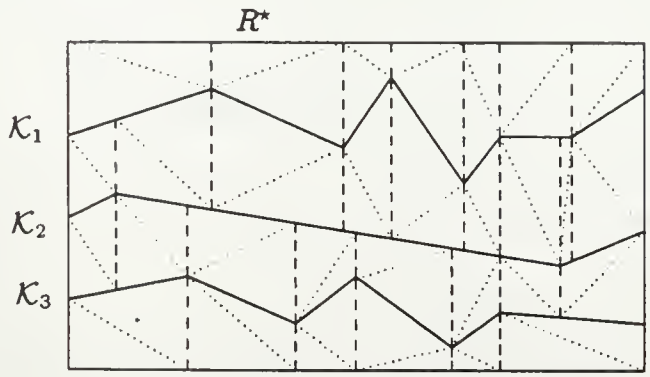


Figure 15: Triangulation of $\mathcal{K}^\star$

Note that K_{3i-1} lies between the levels $3i\frac{n}{r} - \frac{3}{2} \cdot \frac{n}{r}$ and $3i\frac{n}{r} + \frac{1}{2} \cdot \frac{n}{r}$, and K_{3i+1} lies between the levels $3i\frac{n}{r} - \frac{1}{2} \cdot \frac{n}{r}$, $3i\frac{n}{r} + \frac{3}{2} \cdot \frac{n}{r}$. Therefore Π_i lies between the levels $3i\frac{n}{r} \mp \frac{3}{2} \cdot \frac{n}{r}$, which shows that $\Pi_1, \ldots, \Pi_{r/3}$ forms an $\frac{3n}{2r}$-approximate leveling of $\mathcal{A}(\mathcal{L})$. By applying Suri's

algorithm [Su] of computing a minimum link path in the simply connected region, lying between K_{3i-1} and K_{3i+1}, we obtain a path Π'_i, such that $|\Pi'_i| \leq |\Pi_i|$. Since Π'_i lies between K_{3i-1} and K_{3i+1}, it is also a $\frac{3n}{2r}$-approximate $\frac{3in}{r}$-level. Hence, we have

Corollary 6.6 *Given a set $\mathcal{L}$ of n lines and a parameter $1 < r < n$, we can compute a set of $\frac{3n}{2r}$-approximate levels of $\mathcal{A}(\mathcal{L})$, $\mathcal{K} = \{\Pi'_1, \ldots, \Pi'_{r/3}\}$ in time $O(nr \log n \log^{\omega} r)$ with the property that $\sum_{i=1}^{r/3} |\Pi'_i| = O(r^2)$.*

□

Matoušek has also proved that

Lemma 6.7 ([Maa]) *There are at most $O(nr)$ intersections between $\mathcal{K}$ and $\mathcal{L}$.*

□

In view of Lemma 6.7, $\mathcal{K}$ can be decomposed into $O(r^2)$ edges, each intersecting at most $O(\frac{n}{r})$ lines of $\mathcal{L}$. Next, construct the vertical decomposition $\mathcal{K}^*$ of $\mathcal{K}$ by drawing a vertical line from every vertex of $\mathcal{K}$ in both directions until it meets an edge of $\mathcal{K}$ or $\mathbf{R}$ (see figure 15). Since every vertical edge added to $\mathcal{K} \cup \mathbf{R}$ lies within $\frac{3n}{r}$ levels, it intersects at most $\frac{3n}{r}$ lines of $\mathcal{L}$. Therefore, every trapezoid of $\mathcal{K}^*$ intersects at most $O(\frac{n}{r})$ lines of $\mathcal{L}$. Finally, partition each trapezoidal cell of $\mathcal{K}^*$ into two triangles. Hence, we can obtain the main and final result of the paper.

Theorem 6.8 *Given a set $\mathcal{L}$ of n lines in the plane, and a parameter $1 < r < n$, we can decompose the enclosing rectangle $\mathbf{R}$ into $O(r^2)$ triangles in time $O(nr \log n \log^{\omega} r)$, for some constant $\omega < 3.33$, so that no triangle meets more than $O(\frac{n}{r})$ lines of $\mathcal{L}$.*

□

7 Coping with Degenerate Cases

In this section we show how to modify our partitioning algorithm so that it also works in degenerate cases, when more than two lines of $\mathcal{L}$ are concurrent, or more than one intersection point of lines in $\mathcal{L}$ lie on the same vertical line. To facilitate these modifications, we first need to redefine some of the terminology, introduced in Section 2 and Section 4.

(i) If $t \geq 2$ lines of $\mathcal{L}$ pass through a point p, then we consider p as $\binom{t}{2}$ intersection points. Therefore, although the number of distinct intersection points in $\mathcal{L}$ can be less than $\binom{n}{2}$, the sum of their "weights", as just defined, is still $\binom{n}{2}$.

(ii) The level of a vertex of $\mathcal{A}(\mathcal{H})$ can no longer be uniquely defined, so we redefine the k-level of $\mathcal{A}(\mathcal{H})$ to be the polygonal path formed by the closure of the (open) edges of $\mathcal{A}(\mathcal{H})$ whose level is k. Let $p_0, \ldots, p_m$ be the vertices of a k-level of $\mathcal{A}(\mathcal{H})$. Let w_i denote the weight of p_i, and for any $\delta < m$, let p_{i_j} be the vertex of a k-level such that $\sum_{t=0}^{i_j-1} w_t < j\delta \leq \sum_{t=0}^{i_j} w_t$. We now define the *$\delta$-simplified k-level* to be the polygonal path connecting p_0 to p_{i_1}, p_{i_1} to $p_{i_2}, \ldots, p_{i_s}$ to p_m, for $s = \lfloor \frac{\sum_i w_t}{\delta} \rfloor$, concatenated with the left and right rays of the k-level incident to p_0 and p_m respectively.

(iii) For a convex quadrilateral $\mathcal{Q}$, we use $\mathcal{L}$ to denote the set of lines passing through the *interior* of $\mathcal{Q}$, and we let K denote the total weight of the intersection points contained in the interior of $\mathcal{Q}$.

(iv) The set A (resp. B) of left (resp. right) endpoints of the lines in $\mathcal{L}$, defined in Section 2, now becomes a multi-set, because many lines can have a common endpoint. If two lines ℓ_i and ℓ_j have a common left (resp. right) endpoint x and ℓ_i lies counterclockwise (resp. clockwise) from ℓ_j, when directed from x into $\mathcal{Q}$, then $\pi(i) < \pi(j)$ (resp. $\sigma(i) > \sigma(j)$). In other words, A and B are ordered in the way they should be, if we shrink $\mathcal{Q}$ slightly.

(v) Finally, the quantity δ_i defined in Section 4 denotes the number of pseudo edges whose relative interior intersects ℓ_i. Similarly we define the quantities δ_i^{xy}.

Next we briefly describe the modifications in our algorithm and in its analysis, required to make them work in degenerate cases as well; we leave it for the reader to fill in the details.

Observe that in Section 3.1 we actually count the number of inversions to compute K_{rr}, K_{br} and K_{gr}, which gives the total weight of the intersection points (not the number of distinct intersection points) contained in $\mathcal{Q}$, so this part of the algorithm does not require any modification. However, now it is not always possible to find a vertical line that, for a given $k \leq K$, has exactly k intersection points in $\mathcal{Q}$ to its left (cf. Section 3.2). Instead, we find the rightmost vertical line having $\leq k$ intersections in $\mathcal{Q}$ to its left. If we interpret the order of the lines of $\mathcal{L}$ along a vertical line λ as the order that would result by moving λ slightly to the left, then it is easily checked that the procedure described in Section 3.2, with obvious and trivial modifications, would produce the desired line. We now apply this procedure, in the final step of our general partitioning algorithm (cf. Section 4.2), to partition a cell $\mathcal{F}_\alpha$ into subcells, each containing $\leq \zeta\sqrt{K}$ intersection point in its interior.

The algorithm of Section 4.1 remains the same. Since many elements in A or B can have the same value, two or more pseudo edges can have a common endpoint (see figure 16a), or even completely overlap; however, we will regard overlapping pseudo edges as a single edge. It can be checked that our new conventions regarding A and B imply that a line $\ell_i \in \mathcal{L}$ intersects at least $|\pi(i) - \sigma(i)|$ lines inside $\mathcal{Q}$, that is, Lemma 4.1 still holds, which in conjunction with the new definitions of K and δ implies that Lemma 4.2 also continues

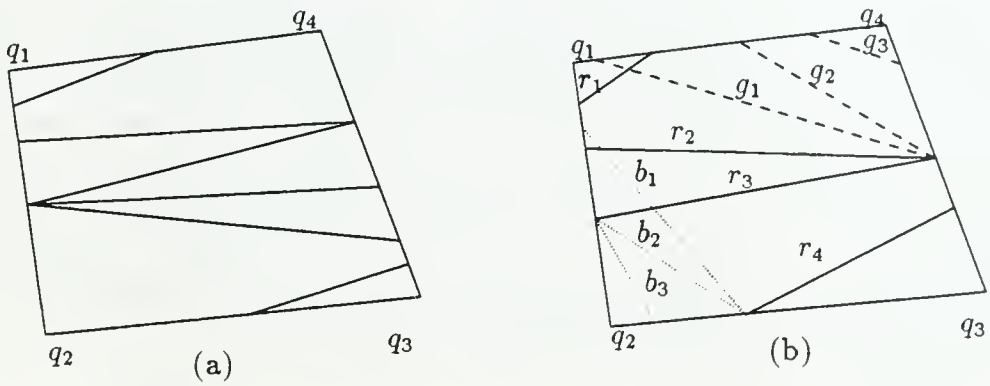


Figure 16: Cells in degenerate cases: (a) special case; (b) general case

to hold. Lemma 4.4 is trickier to adjust. We redefine μ_i as the number of lines passing through $\mathcal{Q}_i$ and not having an endpoint whose rank in A or in B is between $(i-1)\zeta+1$ and $i\zeta$. With some care, the proof can be modified to yield the same bound on μ_i.

Our general algorithm described in Section 4.2 also does not change except that in the final step we now use the modified algorithm mentioned above to partition the cells $\mathcal{F}_\alpha$ into $\lceil \frac{K_\alpha}{\zeta\sqrt{K}} \rceil$ cells, each containing at most $\zeta\sqrt{K}$ intersection points in its interior. Note that now we can have some triangular cells that do not have an edge lying along $\partial\mathcal{Q}$ but only have a vertex lying on $\partial\mathcal{Q}$; these cells are also considered as boundary cells. We may even have some zero-area cells, caused by overlapping pseudo edges, but we discard these cells. In view of our convention regarding the weights of intersection points and the order of the lines along $\partial\mathcal{Q}$, it is easy to see that the charging scheme of Lemma 4.7 still works. (More specifically, we charge a red-green intersection to a red-green cell $\mathcal{F}_{ij}$ if the rank of the right endpoint of the red line is between $(j-1)\zeta+1$ and $j\zeta$ and the rank of the appropriate endpoint of the green line is between $(i-1)\zeta+1$ and $i\zeta$, and similarly for red-blue intersections.) Moreover, using the same argument as in Section 4.3 we can prove Lemma 4.10 and Lemma 4.11 in this degenerate setting because we are not counting those pseudo lines that intersect ℓ at their endpoints. Finally, it can be shown that Lemma 4.9, Lemma 4.12 and Lemma 4.13 also hold, because, although a line ℓ_i can meet the boundary of many cells, it meets the interior of exactly δ_i+1 cells. Lemma 4.14 can also be shown to hold, using our notational convention in degenerate cases.

As for the second phase of our algorithm, Lemma 6.1 and Lemma 6.2 are not affected by degeneracies in $\mathcal{L}$. However, computing the graphs $\mathcal{G}_k$ now becomes slightly more difficult because the level of a vertex v of $\mathcal{G}$ may be undefined (that is, when v is a "heavy-weighted" vertex of $\mathcal{A}(\mathcal{L})$ as well). But for each vertex v of $\mathcal{G}$, we can compute the lines of $\mathcal{L}$ passing through it in time $O(nr\log n)$, because a line passing through a vertex v either lies in the interior of a triangle adjacent to v, or contains on of the edges adjacent to v. Moreover for each edge incident to v, we can compute, in time $O(nr\log n)$, its level in a sufficiently small neighborhood of v. Therefore, we can still propagate the levels from left to right

along the edges of $\mathcal{G}$, and can determine the levels crossed by each edge of $\mathcal{G}$. Thus, we can partition $\mathcal{G}$ into $\mathcal{G}_1, \ldots, \mathcal{G}_r$ and obtain an $\frac{n}{2r}$-approximate leveling of $\mathcal{A}(\mathcal{L})$. It can be checked that the technical results in [EW] and [Maa] can be extended to the degenerate case, which in turn implies that Lemma 2.3 and Lemma 6.5 still hold if we follow our new conventions regarding simplified levels and the weights of intersection points. It is also easy to check that the proof of Lemma 6.7 does not require the lines of $\mathcal{L}$ to be in general position.

Hence, we can conclude that Theorem 6.8 holds, with appropriate modifications as discussed above, even if the lines are not in general position.

8 Conclusions

In this paper we presented a deterministic algorithm that, given a set $\mathcal{L}$ of n lines and a parameter $1 < r < n$, partitions the plane into $O(r^2)$ triangles, each of which meets at most $O(\frac{n}{r})$ lines of $\mathcal{L}$. Although we showed that our algorithm is optimal up to a poly-log factor, there are some questions that still remain unanswered:

(i) The time complexity of our algorithm has an extra $\log^{\omega} r$ factor. The value of ω depends on the constants appearing in the bounds of $\max m_i$ and $\sum_{i=1}^{M} m_i$ in Theorem 4.15. We believe the values of these constants can be improved by doing a more careful analysis, so a natural question is to determine the best possible values of these constants.

(ii) As mentioned in the introduction, we conjecture that the second phase of our algorithm is redundant, i.e. the first phase itself or some appropriate variant of it produces $O(r^2)$ triangles; so far we have not been able to prove this conjecture.

(iii) One drawback of our algorithm is that its space complexity is $O(nr)$. This is because at every level of recursion we need to determine the lines passing through each cell. Can the space complexity be reduced to $O(n + r^2)$?

(iv) The $\Omega(nr)$ lower bound applies only if we want to report the lines passing through each triangle. No non-trivial lower bound is known if these crossing need not be reported. A challenging open problem is to establish a similar lower bound in this case too.

Besides the above open problems related to our algorithm, there are several other open problems related to the general partitioning problem.

(i) The most challenging open problem is to generalize our algorithm to three dimensions. That is, for a given set H of n planes in $\mathbb{R}^3$ and a parameter $1 \leq r < n$,

how fast can one partition space into $O(r^3)$ tetrahedra so that no tetrahedron meets more than $O(\frac{n}{r})$ planes of H? The algorithm of [CF] yields such a partitioning in time $O(n^{10})$. Since the number of plane-tetrahedron crossings is $\Theta(nr^2)$ in the worst case, an ideal solution would be an algorithm whose running time is close to $O(nr^2)$.

(ii) Our algorithm relies on the fact that the objects are straight lines, so it is still an open problem whether a similar algorithm exists for curves. Note that the algorithm of Chazelle and Friedman [CF] works even for curves, but their construction is very inefficient. Moreover, in the case of curves, the techniques of [CF], as well as the random sampling technique, both give a slightly weaker bound, namely, they partition the plane into $O(r^2)$ simply-shaped regions, each meeting at most $O(\frac{n}{r}\log r)$ curves (instead of $O(\frac{n}{r})$).

(iii) Suppose we have a collection of m blue lines and a collection of n red lines. For a given parameter r, how fast can one partition the plane into $O(r^2)$ triangles, so that each triangle meets $O(\frac{m}{r})$ blue lines and $O(\frac{n}{r})$ red lines? Note that the random sampling technique easily yields a similar partitioning, with each triangle meeting $O(\frac{m}{r}\log r)$ blue lines and $O(\frac{n}{r}\log r)$ red lines.

(iv) Suppose we have a collection $\mathcal{G}$ of n segments which intersect at K points. The random sampling technique shows that, for a given parameter r, we can partition the plane into $O(r + \frac{Kr^2}{n^2})$ triangles so that no triangle meets more than $O(\frac{n}{r}\log r)$ segments of $\mathcal{G}$. Can our algorithms be extended to yield a deterministic algorithm that produces that many triangles, each meeting $O(\frac{n}{r})$ segments, and runs in time close to $O(n + \frac{Kr}{n})$ (the lower bound on triangle-segment crossing in this case is $\Omega(n + \frac{Kr}{n})$).

Acknowledgements

I would like to thank my advisor Micha Sharir for encouraging me to work on this problem, for several valuable discussions, and for reading earlier versions of this paper which helped a lot in improving its quality.

References

[Aga] P. K. Agarwal, Partitioning arrangements of lines II: Applications, Technical Report 447, Dept. Computer Science, New York University, May 1989.

[Agb] P. K. Agarwal, Ray shooting and other applications of spanning trees with low stabbing number, *Proceedings* 5th *Annual Symposium on Computational Geometry*, 1989, pp. 315-325.

[Agc] P. K. Agarwal, Intersection and decomposition algorithms for arrangements of curves in the plane, PhD Thesis, in preparation.

[AS] P. K. Agarwal and M. Sharir, Red-blue intersection detection algorithms with applications to motion planning and collision detection, *Proceedings* 4th *Annual Symposium on Computational Geometry*, 1988, pp. 70-80 (Also to appear in *SIAM J. Computing*).

[AKS] M. Ajtai, J. Komlos and E. Szemerédi, Sorting in $c \log n$ parallel steps, *Combinatorica* 3 (1983), pp. 1-19.

[Ba] K. E. Batcher, Sorting networks and their applications, *Proceedings AFIPS Spring Joint Summer Computer Conference* 32 (1968), pp. 307-314.

[Cha] B. Chazelle, Reporting and counting segment intersections, *J. Computer and Systems Sciences* 32 (1986), pp. 156-182.

[Chb] B. Chazelle, Polytope range searching and integral geometry, *Proceedings* 28th *Annual IEEE Symposium on Foundations of Computer Science*, 1987, pp. 1-10.

[Chc] B. Chazelle, Private Communication.

[CE] B. Chazelle and H. Edelsbrunner, An optimal algorithm for intersecting line segments in the plane, *Proceedings* 29th *Annual IEEE Symposium on Foundations of Computer Science*, 1988, pp. 590-600.

[CF] B. Chazelle and J. Friedman, A deterministic view of random sampling and its use in geometry, *Proceedings* 29th *Annual IEEE Symposium on Foundations of Computer Science*, 1988, pp. 539-549.

[CW] B. Chazelle and E. Welzl, Quasi optimal range searching in spaces with finite VC–dimensions, to appear in *Discrete and Computational Geometry*, 1989.

[Cla] K. Clarkson, A probabilistic algorithm for the post office problem, *Proccedings* 17th *Annual ACM Symposium on Theory of Computing*, 1985, pp. 75-84.

[Clb] K. Clarkson, New applications of random sampling in computational geometry, *Discrete and Computational Geometry* 2 (1987), pp. 195-222.

[Clc] K. Clarkson, Applications of random sampling in computational geometry II, *Proceedings* 4th *Annual Symposium on Computational Geometry*, 1988, pp. 1-11.

[CEGSW] K. Clarkson, H. Edelsbrunner, L. Guibas, M. Sharir and E. Welzl, Combinatorial complexity bounds for arrangements of curves and surfaces, *Proceedings* 29th *Annual IEEE Symposium on Foundations of Computer Science*, 1988, pp. 568-579.

[CS] K. Clarkson and P. Shor, Algorithms for diametric pairs and convex hulls that are optimal, randomized and incremental, *Proceedings* 4th *Annual Symposium on Computational Geometry*, 1988, pp. 12-17.

[CTV] K. Clarkson, R. E. Tarjan and C. J. Van Wyk, A fast Las Vegas algorithm for triangulating a simple polygon, *Proceedings* 4th *Annual Symposium on Computational Geometry*, 1988, pp. 18-22.

[Co] R. Cole, Slowing down sorting networks to obtain faster sorting algorithms, *J. ACM* 31 (1984), pp. 200-208.

[CSSS] R. Cole, J. Salowe, W. Steiger and E. Szemerédi, Optimal slope selection, *Proceedings* 15th *International Colloquium on Automata, Languages and Programming*, Lecture Notes in Computer Science, vol. 318, Springer-Verlag, New York, 1988, pp. 133-146.

[CSY] R. Cole, M. Sharir and C. K. Yap, On k-hulls and related problems, *SIAM J. Computing* 16 (1987), pp. 61-77.

[Ed] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, Heidelberg, 1987.

[EGH*] H. Edelsbrunner, L. Guibas, J. Hershberger, R. Seidel, M. Sharir, J. Snoeyink and E. Welzl, Implicitly representing arrangements of lines or segments, *Proceedings* 4th *Annual Symposium on Computational Geometry*, 1988, pp. 56-69. (Also to appear in *Discrete and Computational Geometry.*)

[EGS] H. Edelsbrunner, L. Guibas and M. Sharir, The complexity of many faces in arrangements of lines or of segments, *Proceedings* 4th *Annual Symposium on Computational Geometry*, 1988, pp. 44-55. (Also to appear in *Discrete and Computational Geometry.*)

[EW] H. Edelsbrunner and E. Welzl, Constructing belts in two-dimensional arrangements with applications, *SIAM J. Computing* 15 (1986), pp. 271-284.

[GOS] L. Guibas, M. Overmars and M. Sharir, Ray shooting, implicit point location, and related queries in arrangements of segments, Technical Report 433, Dept. Computer Science, New York University, March 1989.

[HW] D. Haussler and E. Welzl, ϵ-nets and simplex range queries, *Discrete and Computational Geometry* 2 (1987), pp. 127-151.

[MS] H. Mairson and J. Stolfi, Reporting and counting intersections between two sets of line segments, *Theoretical Foundations of Computer Graphics and CAD*, ed. R. Earnshaw, NATO ASI Series, F40, Springer Verlag, Berlin, 1988, pp. 307-325.

[Maa] J. Matoušek, Construction of ϵ-nets, *Proceedings* 5th *Annual Symposium on Computational Geometry*, 1989, pp. 1-10.

[Mab] J. Matoušek, Constructing spanning trees with low crossing numbers, manuscript, 1989.

[Me] N. Megiddo, Applying parallel computation algorithms in design of serial algorithms, *J. ACM* 30 (1983), pp. 852-865.

[RSa] J. Reif and S. Sen, Optimal randomized parallel algorithms for computational geometry, *Proceedings* 16th *International Conference on Parallel Processing*, 1987.

[RSb] J. Reif and S. Sen, Polling: A new randomized sampling technique for computational geometry, *Proccedings* 21st *Annual ACM Symposium on Theory of Computing*, 1989.

[Su] S. Suri, A linear algorithm for minimum link paths inside a simple polygon, *Computer Vision, Graphics and Image Processing* 35 (1986), pp. 99-110.

[Wea] E. Welzl, More on k-sets of finite sets in the plane, *Discrete and Computational Geometry* 1 (1986), pp. 83-94.

[Web] E. Welzl, Partition trees for triangle counting and other range searching problems, *Proceedings* 4th *Annual Symposium on Computational Geometry*, 1988, pp. 23-33.

[Wo] G. Woeginger, Epsilon-nets for half planes, Technical Report B-88-02, Dept. of Mathematics, Free University, Berlin, March 1988.

NYU COMPSCI TR-446
Agarwal, Pankaj K
Partitioning arrangements
of lines: I.
c.1

NYU COMPSCI TR-446
Agarwal, Pankaj K
Partitioning arrangements
of lines: I.
c.1

DATE DUE	BORROWER'S NAME

LIBRARY
N.Y.U. Courant Institute of

This book may be kept

FOURTEEN DAYS

A fine will be charged for each day the book is kept overtime.

GAYLORD 142

PRINTED IN U.S.A.